GNU Rush – a restricted user shell

Sergey Poznyakoff

Short Contents

Table of Contents

1 Introduction

GNU Rush is a Restricted User Shell, designed for sites that provide limited remote access to their resources, such as svn or git repositories, scp, or the like. Using a sophisticated configuration file, GNU Rush gives you complete control over the command lines that users execute, as well as over the usage of system resources, such as virtual memory, CPU time, etc.

2 Operation

GNU Rush is usually installed as a user shell. When a user connects to the server (e.g. by using using SSH protocol), the shell binary, `rush`, is executed. GNU Rush must be called with exactly two arguments: the `-c` command line option and a command line to be executed on the host machine[1]. If wrong arguments are supplied, the shell aborts.

The third argument to `rush` supplies a command line to be executed. This command line along with the password database entry for the user who executes `rush` are said to form a *request*.

After startup, `rush` reads a set of *rules* from its configuration file. Each rule consists of conditions and actions. *Conditions* are used to match the rule with the request. They can include regular expression matching with entire command line or particular fields thereof, user name or group comparisons, etc. If all conditions match the request, actions are executed. *Actions* allow to:

- Modify the command line;
- Impose resource limits;
- Set umask;
- Change current working directory;
- Modify the execution environment;
- Run command in a special root directory ('`chroot`').

Finally, after all actions have been executed successfully, `rush` executes the requested command. Notice, that the resulting command line is not necessarily the same as was supplied to `rush` via the `-c` option.

A special kind of rules, called *fall-through* ones, is provided. Fall-through rules differ from other rules in that they do not execute the command. After all actions in a fall-through rule have been executed, GNU Rush continues to search for another matching rule in its configuration and applies it, if found. Fall-through rules are useful to set default values for subsequent rules.

---

[1] Starting from version 1.6, it is possible to use GNU Rush for interactive shell sessions. See Section 4.2.9 [Interactive], page 21, for more information about it.

3 Quick Start

To give you the feel of GNU Rush possibilities, let's consider the following configuration file rule:

```
rule sftp
  # Conditions:
  command ^.*/sftp-server
  uid >= 100
  # Actions:
  transform[0] s,.*,bin/sftp-server,
  umask 002
  chroot ~
  chdir /
```

The first clause, `rule`, defines a new rule. Its argument serves as a rule tag, used for diagnostic messages and for accounting.

Lines beginning with '`#`' are comments, they are intended for a human reader and are ignored by `rush`.

The two statements that follow the comment, `command` and `uid`, define conditions that must be met for this rule to become active. The `command` statement introduces a regular expression to match with the command line. In this example, the command line must begin with '`/sftp-server`', optionally preceded by arbitrary directory components.

The `uid` statement tells that this rule applies only to users whose UIDs are greater than or equal to 100.

Subsequent clauses define actions associated with this rule.

The `transform[0]` clause contains instructions on how to modify the first argument of the command line (i.e. the command name). These instructions are in the form of `sed` replace expression (see Section 4.2.2 [Transformations], page 11). The expression in our example instructs GNU Rush to replace the command name with '`bin/sftp-server`'[1].

The `umask` clause sets the file creation mask.

The `chroot` clause instructs GNU Rush to chroot to the user home directory before executing the command.

Finally, the `chdir` statement sets the directory to change to after installing the chroot.

---

[1] In this particular case, the `set` statement, introduced in GNU Rush version 1.6, is probably more appropriate:

```
set[0] bin/sftp-server
```

4 Configuration File

The configuration file is called `rush.rc` and is located in `/usr/local/etc` by default.[1].

The configuration file is read and parsed right after start up. Any errors occurred in parsing are reported using `syslog` facility 'authpriv' and priority 'notice'. When run in 'test' mode, all diagnostics is displayed on standard error output. See Chapter 7 [Test Mode], page 37, for a detailed description of ways to debug and test your configurations.

Before parsing, `rush` checks the ownership and permissions of the configuration file for possible security breaches. The configuration file is considered unsafe if any of the following conditions are met:

1. It is not owned by root.
2. It is group writable.
3. It is world writable.
4. It resides in a group writable directory.
5. It resides in a world writable directory.
6. It is a symbolic link to a file residing in a group or world writable directory.

If the file is considered unsafe, `rush` rejects it and aborts execution.

Any of these tests can be disabled using the `--security-check` option (see [–security-check], page 39).

4.1 Syntax

Configuration file consists of statements and comments.

A *comment* is any line whose first non-whitespace character is '#'. Empty lines and comments are ignored.

A *statement* consists of a keyword and optional argument, separated by any amount of whitespace. Depending on the keyword, the statement may treat its argument as a single value or as multiple values. For example, the `command` instruction takes a single value as its argument, so parsing the statement

```
command ^scp -t /incoming/
```
results in keyword 'command' and value '^scp -t /incoming/'.

If the keyword requires multiple values, its argument is split into words using the following algorithm:

1. Any sequence of one or more non-whitespace characters is a word.
2. Any sequence of characters enclosed in single ('') or double-quotes ('"') is a word.

---

[1] The exact location of the configuration file is defined when configuring the package. See the file `INSTALL` in the GNU Rush source directory for more information

3. Words are separated by any amount of white space.

4. If the keyword expects s-expressions (see [s-expression], page 13), these are treated as words, even if they contain white space.

Arguments, obtained as a result of rules (1) and (2) are subject to *backslash interpretation*, during which the following *escape sequences* are replaced with single characters, as described in the table below:

Sequence	Replaced with
\a	Audible bell character (ASCII 7)
\b	Backspace character (ASCII 8)
\f	Form-feed character (ASCII 12)
\n	Newline character (ASCII 10)
\r	Carriage return character (ASCII 13)
\t	Horizontal tabulation character (ASCII 9)
\v	Vertical tabulation character (ASCII 11)

Table 4.1: Backslash escapes

Any escape sequence not listed in this table is replaced with its second character.

Statements are delimited by newline characters. Length of a statement line is not limited. To improve readability, long statements may be split over several lines by using backslash ('\') as a last character on line. Thus, the following statement:

```
usage-error Contact your\
 system administrator
```

is equivalent to:

```
usage-error Contact your system administrator
```

4.1.1 Notes on Quoted Strings

The syntax of GNU Rush configuration file was designed so as to implement minimum amount of syntactic mark up. Most statements treat their argument as a single value, even if it contains embedded white space. However, leading and trailing whitespace is always removed. Consider, for example, the following statement[2]:

```
match[1] ^/sources/[^ ]+\.git$
```

Here, the argument is '`^/sources/[^ ]+\.git$`'. Note, that you must not quote it, because quotation marks would be considered part of the argument.

There are, however, statements that take several arguments. In these statements, arguments that contain embedded white space must be quoted.

---

[2] See [match], page 10, for information about `match` statement.

For example, in the statement below[3] the second argument is a single space character. It is quoted to prevent it from being treated as a delimiter:

    map[0] /etc/passwd.rush " " ${user} 1 7

Notice also, that arguments to these statements are subject to backslash interpretation (see Table 4.1).

The table below lists all statements that take multiple arguments, with cross references to more in-depth explanations in the body of the manual.

user   See Section 4.2.1 [Conditions], page 9.

group   See Section 4.2.1 [Conditions], page 9.

transform *pattern expr*
transform[*n*] *pattern expr*
              See Section 4.2.2.3 [transform], page 13.

map   See Section 4.2.2.4 [map], page 14.

env   See Section 4.2.4 [Environment], page 17.

regexp   See Section 4.5 [Regex], page 27.

include-security
              See [include-security], page 25.

4.2 Rule

The `rule` statement configures a GNU Rush rule. This is a *block* statement, which means that all statements located between it and the next `rule` statement (or end of file, whichever occurs first) modify the definition of the rule.

The syntax of the `rule` statement is:

rule *tag*   [Configuration]
The *tag* argument is optional. If it is given, it supplies a *tag* for the rule, i.e. a (presumably unique) identifier, which is used to label this rule. `Rush` uses this tag in its diagnostic messages. For rules without explicit *tag*, `Rush` supplies a default tag, which is constructed by concatenating '`#`' character and the ordinal number of rule in the configuration file, in decimal notation. Rule numbering starts from '`1`'.

The following sub-sections describe statements that can be used within a `rule`.

4.2.1 Conditions

These statements define conditions that are used to match the rule with the request. A rule may contain any number of conditions. All conditions are tested in order of their appearance in the rule and are tied together using boolean shortcut '`and`' evaluation: if any of them yields false, the rest is not evaluated and control is transferred to the subsequent rule.

---

[3] See Section 4.2.2.4 [map], page 14, for a description of `map` statement.

`command` *regex*   [Rule Config]
>   True, if the current command line matches regular expression *regex*.
>
>   For example:
>
>   command ^scp (-v)?-t /incoming/(alpha|ftp)
>
>   By default, POSIX extended regular expressions are used. This, however
>   can be changed using `regex` statement (see Section 4.5 [Regex], page 27).

`match[n]` *regexp*   [Rule Config]
>   True, if *n*th *word* from the command line matches regular expression
>   *regexp*. Notice, that square brackets form part of the statement syntax.
>   A special value '`$`' can be used instead of *n* to denote the last word. Unless
>   changed by previous `regex` statement (see Section 4.5 [Regex], page 27),
>   POSIX extended regular expressions are used.
>
>   The command line is split into words using the same rules as used in
>   `/bin/sh`.
>
>   For example, the condition below yields true if the last argument is an
>   absolute path name:
>
>   match[$] ^/.*

`argc` *op num*   [Rule Config]
>   Compare the number of command line arguments to *num*. The compar-
>   ison operator is given by *op*, which can be one of the following: '`=`' (or
>   '`==`'), '`!=`', '`<`', '`<=`', '`>`', '`>=`'.
>
>   For example, the following condition matches if the number of arguments
>   is less than 3:
>
>   argc < 3

`uid [`*op*`]` *user-id*   [Rule Config]
>   Compare current UID to *user-id*. The latter may be either a numeric UID
>   or a name of an existing user.
>
>   The comparison operator is given by optional *op*, which can be one of the
>   following: '`=`' ('`==`'), '`!=`', '`<`', '`<=`', '`>`', '`>=`'. If *op* is not given, equality
>   ('`==`') is assumed.
>
>   Examples:
>
>   uid smith

`gid` *op group-id*   [Rule Config]
>   Compare current GID to *group-id*, which is either a numeric value or a
>   name of an existing group.
>
>   The comparison operator is given by *op*, which can be one of the following:
>   '`=`' ('`==`'), '`!=`', '`<`', '`<=`', '`>`', '`>=`'. If *op* is not given, equality ('`==`') is
>   assumed.

`user` *names*   [Rule Config]
>   Argument is a whitespace-separated list of user names. This condition
>   yields true, if the user name matches one of the listed names. String
>   comparisons are case-sensitive.

group *names*   [Rule Config]
> Argument is a whitespace-separated list of group names. This condition
> yields true, if the the name of any group the user is a member of matches
> one of listed names. String comparisons are case-sensitive.

> For example, to match users from groups 'admin' and 'root':
>> ```
>> group admin root
>> ```

Each condition allows for a negated form, by placing an exclamation sign
between the condition keyword and expression. For example:

command ^scp
> True, if the command line begins with 'scp'.

command ! ^scp
> True if the command line does not begin with 'scp'.

4.2.2 Transformations

Special actions that allow to rewrite command line are called *transforma-
tions*. GNU Rush supports five kinds of transformations: 'set', 'transform',
'delete', 'map'. All of them operate on command line split into *words*. Ad-
ditionally, 'set' and 'transform' can operate on the entire command line.

Rush performs word splitting using the same rules as sh. Transformation
actions refer to words by their *index*. Three kinds of indexes are supported.
First of all, a word may be referred to by its ordinal number in the com-
mand line. The command name itself has index '0'. For example, given the
command line:
```
/bin/scp -t /upload
```
one gets:

Index	Value
0	/bin/scp
1	-t
2	/upload

Negative indexes can also be used. They refer to words in the backward
order, as illustrated in the following table:

Index	Value
-1	/upload
-2	-t
-3	/bin/scp

Finally, the last word may be referred to as '$', and the command name
itself as '^'. There is a subtle difference between '0' and '^'. The notation
'^' refers to the name of the program that rush will execute at the end
of the matching rule, whereas the notation '0' refers to the 0th argument
that will be passed to that program ('argv[0]'). Most of the time the two
values coincide. Unless the rule modifies '^', '0'th word will be used as the

program name. There exist some cases when you need to explicitly set '^'.
See Section 4.2.9 [Interactive], page 21, for a possible use of this feature.

Some of the transformations implement *patterns*. A pattern is a string
which may contain *meta-variables*. Before using, these meta-variables are
expanded using the following rules:

Meta-variable	Expansion
${user}	User name
${group}	Name of the user's principal group
${uid}	UID
${gid}	GID
${home}	User's home directory
${gecos}	User's GECOS field
${program}	Program name ('^')
${command}	Full command line
$0 to $9	Corresponding word from the command line
${N}	Nth word (see above for the allowed values of N)

4.2.2.1 Set

The 'set' transformation allows to replace entire command line, or any of
its arguments, with the given value.

set *pattern*   [Rule Config]
> Command line is replaced with the expansion of *pattern* (see [patterns],
> page 12). E.g.:
>
> ```
> set /bin/echo "Command forbidden: ${command}"
> ```
>
> As another example, the following rule uses transform to ensure that
> /usr/bin/cvs binary is used:
>
> ```
> rule cvs
>   command ^cvs server
>   set[0] /usr/bin/cvs
> ```
>
> In versions of GNU Rush prior to 1.6, it was common to use the
> transform action for this purpose.

set[*n*] *pattern*   [Rule Config]
> Replace nth argument with the expansion of *pattern*. Notice, that square
> brackets are part of the statement syntax.
>
> See [indexing], page 11, for a description of n. See [patterns], page 12, for
> a description of *pattern*. E.g.:
>
> ```
> set[0] /bin/echo
> ```

4.2.2.2 Delete

The 'delete' action deletes the given word, or range of words, from the
command line. It has two forms.

delete[n]   [Rule Config]
> Delete nth word. See [indexing], page 11, for a detailed description of n
> and its syntax. However, see the note below.

delete n m   [Rule Config]
> Delete words from n to m, inclusive. For example, the following action
> removes all arguments, except the command name:
> ```
> delete 1 $
> ```

> Neither form can be used to delete the command name, i.e. '[0]' or '[^]'.

4.2.2.3 Transform

The `transform` action modifies the value of the command line, or any partic-
ular word of it, according to a sed *s-expression*. S-expressions are described
in more detail below (see [s-expression], page 13). This action has several
forms.

transform *expr*   [Rule Config]
> Apply expression *expr* to entire command line. For example, the action
> below adds a `-t` option after the command name:
> ```
> transform s,^[^ ]+,& -t,
> ```

transform *pattern expr*   [Rule Config]
> Expand *pattern* as described in [patterns], page 12, apply *expr* to the
> expansion, and replace the command line with the result.

transform[n] *expr*   [Rule Config]
> Apply expression *expr* to nth *word* from the command line. Notice, that
> square brackets are part of the statement syntax. See [indexing], page 11,
> for a detailed description of n and its syntax.

transform[n] *pattern expr*   [Rule Config]
> Apply expression *expr* to the expanded *pattern* and assign the result to
> nth word. See [patterns], page 12, for a description of patterns and their
> expansion. See [indexing], page 11, for a detailed description of n and its
> syntax.

> The example below replaces the 0th argument with the base name of the
> command, prefixed by a dash:
> ```
> transform[0] ${^} s,.*/,-,
> ```
> For instance, if the command name is '/bin/bash', 'argv[0]' will become
> '-bash'.

The transformation expression, *expr*, is a `sed`-like replace expression of
the form:
```
s/regexp/replace/[flags]
```
where *regexp* is a *regular expression*, *replace* is a replacement for each part
of the input that matches *regexp*. Both *regexp* and *replace* are described in
detail in Section "The 's' Command" in *GNU sed*.

As in `sed`, you can give several replace expressions, separated by a semi-colon.

Supported *flags* are:

'`g`'   Apply the replacement to *all* matches to the *regexp*, not just the first.

'`i`'   Use case-insensitive matching

'`x`'   *regexp* is an *extended regular expression* (see Section "Extended regular expressions" in *GNU sed*).

'`number`'   Only replace the *number*th match of the *regexp*.

Note: the POSIX standard does not specify what should happen when you mix the '`g`' and *number* modifiers. `Rush` follows the GNU `sed` implementation in this regard, so the interaction is defined to be: ignore matches before the *number*th, and then match and replace all matches from the *number*th on.

Any delimiter can be used in lieu of '`/`', the only requirement being that it be used consistently throughout the expression. For example, the following two expressions are equivalent:

```
transform s/one/two/
transform s,one,two,
```

Changing delimiters is often useful when the *regex* contains slashes. For instance, it is more convenient to write `s,/,-,` than `s/\//-/`.

For example, the following rule uses `transform` to ensure that `/usr/bin/cvs` binary is used:

```
rule cvs
  command ^cvs server
  transform[0] s|.*|/usr/bin/cvs|
```

The same effect can be achieved with a `set` statement, as shown in [set-command], page 12.

As a more complex example, consider the following rule:

```
rule svn
  command ^svnserve -t
  transform s|-r *[^ ]*||;s|^svnserve |/usr/bin/svnserve -r /svnroot |
```

This transform expression first removes all occurrences of `-r` option and its arguments from the command line, and then adds its own `-r` option and replaces '`svnserve`' with the full program file name.

4.2.2.4 Map

The '`map`' statement uses file lookup to find a new value for the command line word.

`map[n]` *file delim pattern kn vn*   [Rule Config]
`map[n]` *file delim pattern kn vn default*   [Rule Config]

Arguments are:

n	Argument index, as in [indexing], page 11.
file	Name of the *map file*. It must begin with '/' or '~/'. Before using, the file permissions and ownership are checked using the procedure described in [security checks], page 7.
delim	A string containing allowed field delimiters.
pattern	The value of the lookup key. Before using, it is expanded as described in [patterns], page 12.
kn	Number of the key field in *file*. Fields are numbered starting from 1.
vn	Number of the value field.
default	If supplied, this value is used as a replacement value, when the key was not found in *file*.

The map file consists of *records*, separated by newline characters (in other words, a record occupies one line). Each record consists of fields, separated by delimiters, given in *delim* argument. If *delim* contains a space character, then fields may be delimited by any amount of whitespace characters (spaces and/or tabulations). Otherwise, exactly one delimiter delimits fields.

Fields are numbered starting from 1.

The `map` action works as follows:

1. The *pattern* argument is expanded as described in [patterns], page 12, and the resulting value is used as lookup key.
2. The *file* is scanned for a record whose *kn*th field matches the lookup key.
3. If such a record is found, the value of its *vn*th field is assigned to the *n*th command line word (see [indexing], page 11, for a description of *n*).
4. Otherwise, if *default* is supplied, it becomes the new value of the *n*th word.
5. Otherwise, the *n*th word remains unchanged.

For example, suppose that the file `/etc/passwd.rush` has the same syntax as the system `passwd` file (see Section "passwd" in *passwd(5) man page*). Then, the following statement will replace 'argv[0]' with the value of 'shell' field, using the current user name as a key:

```
map[0] /etc/passwd.rush : ${user} 1 7
```

See also Section 4.2.9 [Interactive], page 21, for another example of using this statement.

4.2.3 System Actions

System actions provide an interface to the operating system.

umask *mask*   [Rule Config]
    Set the umask. The *mask* must be an octal value not greater than '0777'.
    The default umask is '022'.

newgrp *group-id*   [Rule Config]
newgroup *group-id*   [Rule Config]
    Changes the current group ID to *group-id*, which is either a numeric value
    or a name of an existing group.

chroot *dir*   [Rule Config]
    Change the root directory to that specified in *dir*. This directory will be
    used for file names beginning with '/'. A tilde ('~') at the start of *dir* is
    replaced with the user's home directory.

    The directory *dir* must be properly set up to execute the commands. For
example, the following rule defines execution of `sftp-server` in an environ-
ment, chrooted to the user's home directory:

```
rule sftp
  command ^.*/sftp-server
  set[0] bin/sftp-server
  chroot ~
```

For this to work, each user's home must contain the directory `bin` with a
copy of `sftp-server` in it, as well as all directories and files that are needed
for executing it, in particular `lib`.

chdir *dir*   [Rule Config]
    Change to the directory *dir*. The argument is subject to tilde-expansion
    (see `chroot`, above). If both `chdir` and `chroot` are specified, then `chroot`
    is executed first.

limits *res*   [Rule Config]
    Impose limits on system resources, as defined by *res*. The argument con-
    sists of *commands*, optionally separated by any amount of whitespace. A
    command is a single command letter followed by a number, that specifies
    the limit. The command letters are case-insensitive and coincide with
    those used by the shell `ulimit` utility:

Command	The limit it sets
A	max address space (KB)
C	max core file size (KB)
D	max data size (KB)
F	maximum file size (KB)
M	max locked-in-memory address space (KB)
N	max number of open files
R	max resident set size (KB)
S	max stack size (KB)
T	max CPU time (MIN)
U	max number of processes

L   max number of logins for this user (see below)

P   process priority -20..20 (negative = high priority)

For example:

```
limits T10 R20 U16 P20
```

If some limit cannot be set, execution of the rule aborts. In particular, 'L' limit can be regarded as a condition, rather than action. The setting `limit L`*n* succeeds only if no more than *n* `rush` instances are simultaneously running for the same user. This can be used to limit the number of simultaneously open sessions.

The use of 'L' resource automatically enables *forked mode*. See Section 4.2.6 [Accounting and Forked Mode], page 19, for more information about it.

4.2.4 Environment

The `env` action allows to modify the environment in which the program will be executed.

env *args*   [Rule Config]
    Modify the environment.

Its arguments are a whitespace-delimited list of specifiers. Each specifier can contain references to variables from the inherited environment. The reference syntax is the same as in Bourne shell.

The following specifiers are understood:

- (a dash)   Clear the environment. This is understood only when used as a first word in *args*.

-*name*   Unset the environment variable *name*.

-*name*=*val*
            Unset the environment variable *name* only if its value is *val*.

name   Retain the environment variable *name*.

name=*value*
            Define environment variable *name* to have given *value*.

name+=*value*
            Retain variable *name* and append *value* to its value. If no such variable is present in the environment, it is created and *value* is assigned to it. However, if *value* starts with a punctuation character, this character is removed from it before the assignment. This is convenient for using this construct with environment variables like `PATH`, e.g.:

```
PATH+=:/sbin
```

In this example, if `PATH` exists, '`:/sbin`' will be appended to it. Otherwise, it will be created and '`/sbin`' will be assigned to it.

This is roughly equivalent to

```
PATH=$PATH:/sbin
```

name=+*value*

Retain variable *name* and prepend *value* to its value. If no such variable is present in the environment, it is created and *value* is assigned to it. However, if *value* ends with a punctuation character, this character is removed from it before assignment.

4.2.5 Fall-through

A *fall-through* rule is a special rule that does not execute the requested command. When a matching fall-through rule is encountered, `rush` evaluates it and continues scanning its configuration for the next matching rule. Any transformation and `env` actions found in the fall-through rule take effect immediately, which means that subsequent rules will see modified command line and environment. Execution of any other actions found in the fall-through rule is delayed until a usual rule is found.

A fall-through rule is declared using the following statement:

`fall-through`   [Rule Config]
    Declare a fall-through rule.

Usually this statement is placed as the last statement in a rule, e.g.:

```
rule default
  umask 002
  env - HOME USERNAME PATH
  fall-through
```

Fall-through rules provide a way to set default values for subsequent rules. For example, any rules that follow the '`default`' rule shown above, will inherit the umask and environment set there.

One can also use fall-through rules to "normalize" command lines. For example, consider this rule:

```
rule default
  transform[0] s|.*/||;
  fall-through
```

It will remove all path components from the first command line argument. As a result, all subsequent rules may expect a bare binary name as the first argument.

Yet another common use for such rules is to enable accounting (see the next subsection), or set resource limits for the rest of rules:

```
rule default
  limit l1
  fall-through
```

4.2.6 Accounting and Forked Mode

GNU Rush is able to operate in two modes, which we call default and forked. When operating in the default mode, the process image of `rush` itself is over-written by the command being executed. Thus, when it comes to launching the requested command, the running instance of `rush` ceases to exist.

There is also another operation mode, which we call *forked mode.* When running in this mode, `rush` executes the requested command in a subpro-cess, and remains in memory supervising its execution. Once the command terminates, `rush` exits.

One advantage of the forked mode is that it allows to run *accounting,* i.e. to note who is doing what and to keep a history of invocations. The accounting, in turn, can be used to limit simultaneous executions of com-mands (*logins,* in GNU Rush terminology), as requested by 'L' command to `limit` statement (see [L limit], page 17).

The forked mode is enabled on a per-rule basis, for rules that contain either 'L' command in the `limit` statement, or '`acct on`' command:

`acct` *bool*   [Rule Config]
> Turn accounting mode on or off, depending on *bool.* The argument can be one of the following: '`yes`', '`on`', '`t`', '`true`', or '`1`', to enable accounting, and '`no`', '`off`', '`nil`', '`false`', '`0`', to disable it.
>
> Notice, that there is no need in explicit `acct on` command, if you use `limit L`.

The notion '`rule contains`', used above, means that either the rule in question contains that statement, or inherits it from one of the above fall-through rules (see Section 4.2.5 [Fall-through], page 18). In fact, in most cases the accounting should affect all rules, therefore we suggest to enable it in a fall-through rule at the beginning of the configuration file, e.g.:

```
rule default
  acct on
  fall-through
```

If the need be, you can disable it for some of the subsequent rules by placing `acct off` in it. Notice, that this will disable accounting only, the forked mode will remain in action. To disable it as well and enforce default mode for a given rule, use `fork off` statement:

`fork` *bool*   [Rule Config]
> Enable or disable forked mode. This statement is mainly designed as a way of disabling the forked mode for a given rule.

Once the accounting enabled, you can view the list of currently logged in users using `rushwho` command (see Chapter 9 [Rushwho], page 41) and view the history of last logins using `rushlast` command (see Chapter 10 [Rushlast], page 45).

4.2.7 Post-process Notification

`Rush` can be configured to send a *notification* over INET or UNIX sockets, after completing user request. It is done using the `post-socket` statement:

`post-socket` *url*   [Rule Config]
  Notify URL about completing the user request. This statement implies forked mode (see Section 4.2.6 [Accounting and Forked Mode], page 19).

  Allowed formats for *url* are:

  inet://*hostname*[:*port*]
          Connect to remote host *hostname* using TCP/IP. *Hostname* is the host name or IP address of the remote machine. Optional *port* specifies the port number to connect to. It can be either a decimal port number or a service name from `/etc/services`. If *port* is absent, '`tcpmux`' (port 1) is assumed.

  unix://*filename*
  local://*filename*
          Connect to a UNIX socket *filename*.

  For example:
```
    rule default
      post-socket inet://localhost
```

The GNU Rush notification protocol is based on TCPMUX (RFC 1078 (`http://www.rfc-editor.org/rfc/rfc1078.txt`)).

After establishing connection, `rush` sends the rule tag followed by a CRLF pair. The rule tag acts as a service name. The remote party replies with a single character indicating positive ('`+`') or negative ('`-`') acknowledgment, immediately followed by an optional message of explanation, terminated with a CRLF.

If positive acknowledgment is received, `rush` sends a single line, consisting of the user name and the executed command line, separated by a single space character. The line is terminated with a CRLF.

After sending this line, `rush` closes the connection.

The post-process notification feature can be used to schedule execution of some actions after certain rules.

See Section 6.7 [notification example], page 34, for an example of how to use this feature.

4.2.8 Exit rule

An *exit* rule does not execute any commands. Instead, it writes the supplied error message to the specified file descriptor and exits immediately. An exit rule is defined using the following statement:

exit *fd message*   [Rule Config]
exit *message*   [Rule Config]

    Write textual message *message* to a file descriptor, given by the optional argument *fd*. If *fd* is absent, '2' (standard error) is used.

    The *message* argument is subject to backslash interpretation (see Table 4.1).

    For example (note the use of line continuation character):

```
exit \
    \r\nYou are not allowed to execute that command.\r\n\
    \r\nIf you think this is wrong, ask <foo@bar.com> for assistance.\r\n
```

    If *message* begins with a '@' sign, the remaining characters are treated as the name of a predefined error message (see Section 4.7 [Error Messages], page 28). The corresponding message text is retrieved and used instead of *message*. For example:

```
exit @nologin-message
```

    If the characters after '@' do not correspond to any predefined error message name, an error of type 'config-error' is signaled and rush exits.

    If you need to begin your exit message with a '@' sign, duplicate it, as in this example:

```
exit @@Special error message
```

    This example will produce '@Special error message'.

    Exit actions are useful for writing *trap rules*, i.e. the rules that are intended to trap incorrect or prohibited command lines and to return customized reply messages in such cases. Consider the following rule:

```
rule git
  command ^git-.+
  match[1] ^/sources/[^ ]+\.git$
  transform s|.*|/usr/bin/git-shell -c "&"|
```

    It allows to use only those Git repositories that are located under /sources directory[4]. If a user tries to access a repository outside this root, he will be returned a default error message, saying 'You are not permitted to execute this command' (see Section 4.7 [Error Messages], page 28). You can, however, provide a more convenient message in this case. To do so, place the following after the 'git' rule:

```
rule git-trap
  command ^git-.+
  exit fatal: Use of this repository is prohibited.
```

    This rule will trap all git invocations that do not match the 'git' rule.

4.2.9 Interactive Access

Sometimes it may be necessary to allow some group of users limited access to interactive shells. GNU Rush contains provisions for such usage. When rush

---

[4] See Section 6.6 [git], page 34, for a better way to handle Git accesses.

is invoked without `-c` it assumes interactive usage. In this case only rules explicitly marked as interactive are considered, the rest of rules is ignored.

`interactive`   [Rule Config]
>   This statement marks the rule it appears in as interactive. This rule will match only if `rush` is invoked without command line arguments.

>   Unless command line transformations are applied, interactive rule finishes by executing `/bin/sh`. The first word in the command line (`argv[0]`) is normally set to the basename of the command being executed prefixed by a dash sign.

>   Consider the following example:
>    ```
>    rule login
>      interactive
>      group rshell
>      map[^] /etc/rush.shell : ${user} 1 2
>      transform[0] ${program} s,^,-r,
>
>    rule nologin
>      interactive
>      exit You don't have interactive access to this machine.
>    ```

The 'login' rule will match interactive user requests if the user is a member of the group 'rshell'. It uses `/etc/rush.shell` to select a shell to use for that user (see Section 4.2.2.4 [map], page 14). This map file consists of two fields, separated by a colon. If the shell is found, its base name, prefixed with '-r', will be used as 'argv[0]' (this indicates a restricted login shell). Otherwise, the trap rule 'nologin' will be matched, which will output the given diagnostics message and terminate `rush`.

>   To test interactive access, use the `-i` option:
>    ```
>    rush --test -i
>    ```

4.2.10 Localization

GNU Rush is internationalized, which means that it is able to produce log and diagnostic messages in any language, if a corresponding translation file is provided. This file is called a *localization* or *domain* file. To find an appropriate localization file, `rush` uses the following parameters:

locale   *Locale name* is a string that describes the language, territory and optionally, the character set to use. It consists of the language (ISO 639) and country (ISO 3166) codes, separated by an underscore character, e.g. 'en_US' or 'pl_PL'. If a character set is specified, its name follows the country code and is separated from it by a '@' character.

>   There are two special locale names: 'C' or 'POSIX' mean to use the default POSIX locale, and '""' (an empty string), means to use the value of the environment variable LC_ALL as the locale name.

locale_dir   Directory where localization files are located. If not specified, a predefined set of directories is searched for the matching file.

domain   *Text domain* defines the base name of the localization file.

Given these parameters, the name of the full pathname of the localization file is defined as:

`locale_dir/locale/LC_MESSAGES/domain.mo`

GNU Rush produces three kinds of messages:

diagnostics

These are diagnostics messages that GNU Rush produces to its log output (syslog, unless in test mode).

error messages

Messages sent to the remote party when `rush` is not able to execute the request (see Section 4.7 [Error Messages], page 28).

exit messages

These are messages sent to the remote party by `exit` rules (see Section 4.2.8 [Exit], page 20).

These messages use different domain names (and may use different locale directories). The diagnostics and error messages use textual domain 'rush'. The corresponding locale directory is defined at compile time and defaults to *prefix*/`share/locale`, where *prefix* stands for the installation prefix, which is /usr/local, by default.

GNU Rush is shipped with several localization files, which are installed by default. As of version 1.8, these files cover the following locales:

C
POSIX   This is default domain, it is always supported and does not require any special handling. It is roughly equivalent to '`en_US`' (English).

pl   Polish messages.

uk   Ukrainian messages.

If the localization you need is not in this list, visit `http: / / translationproject .org/domain/rush .html`. If it is not there either, consider writing it (see Section "Translators" in *GNU gettext utilities*, for a detailed instructions on how to do that).

Exit messages use custom domain files. It is responsibility of the system administrator to provide and install such files.

4.2.10.1 Localization Directives

The following configuration directives control localization. They are available for use in `rule` statements:

`locale `*`name`*   [Configuration]
>   Sets the locale name. To specify empty locale, use '`""`' as *name* (recall
>   that empty locale name means use the value of the environment variable
>   `LC_ALL` as locale name).

`locale-dir `*`name`*   [Configuration]
>   Sets the name of the locale directory.

`text-domain `*`name`*   [Configuration]
>   Sets the textual domain name.

>   The following configuration fragment illustrates their use:

```
rule l10n
  locale pl_PL
  text-domain rush-config
  fall-through
```

Different users may have different localization preferences. See [per-user l10n], page 26, for a description of how to implement this.

4.2.10.2 Writing Your Localization

You need to write a localization file for your configuration script if it implements exit rules (see Section 4.2.8 [Exit], page 20) and changes user locale (see Section 4.2.10.1 [Localization Directives], page 23).

Preparing localization consists of three stages: extracting exit messages and forming a PO file, editing this file, compiling and installing it. The discussion below describes these stages in detail.

1. Creating a '`po`' file.

   A PO (*Portable Object*) file is a plain text file, containing original messages and their translations for a particular language. See Section "PO Files" in *GNU gettext utilities*, for a description of its format.

   The script `rush-po.awk` serves for extracting messages from the configuration file and forming a PO file. This script is installed in the *prefix*`/share/rush` directory[5], where *prefix* stands for your installation prefix.

   Supposing the package is installed in the default prefix, the following command will create a PO file from your configuration file:

   ```
   awk -f /usr/local/share/rush/rush-po.awk \
           /usr/local/etc/rush.rc > myconf.po
   ```

   Please note, that `rush-po.awk` does not handle `include` directives (see Section 4.3 [Include], page 25). If your configuration file uses these

---

[5] To be precise, it is installed in *dataroot*`/rush`, where *dataroot* is a directory for read-only architecture-independent data, which is set by the `--datarootdir` option to `configure`. Unless that option is used, this directory defaults to *prefix*`/share/rush`

directives, you need to specify each include file in the `rush-po.awk` command line, e.g.:

```
awk -f /usr/local/share/rush/rush-po.awk \
        /usr/local/etc/rush.rc /home/*/.rush > myconf.po
```

2. Editing the PO file

   Open the created PO file with your favorite editor and supply message translations after `msgstr` keywords. Although you can use any editor, capable of handling plain text files, we recommend to use GNU Emacs, which provides a special *po-mode*. See Section "Basics" in *GNU gettext utilities*, for guidelines on editing PO files and using the po-mode.

3. Compiling the PO file

   When ready, the PO file needs be compiled into a MO (*Message Object*) file, which is directly readable by `rush`. This is done using `msgfmt` utility from GNU gettext:

```
msgfmt -o myconf.mo myconf.po
```

   See Section "msgfmt Invocation" in *GNU gettext utilities*, for a detailed description of the `msgfmt` utility.

   After creating the MO file, copy it into appropriate directory. It is important that the installed MO file uses the naming scheme described in [mo-name], page 23.

4.3 Include

The `include` statement forces inclusion of the named file in that file location:

include *file* [Configuration]
Include file *file*

If *file* starts with a tilde character, followed by a slash ('~/'), these two characters are replaced with the full path name of current user's home directory.

If *file* is a directory, that directory is searched for a file whose name coincides with the current user name. If such a file is found, it is included.

In any case, if the file named by *file* (after tilde expansion) does not exist, no error is reported, and parsing of the configuration file continues.

Before including the file, `rush` checks if it is secure, using the same rules as for the main configuration file (see [security checks], page 7). The exact list of checks can be tuned using the `include-security` statement:

include-security *list* [Configuration]
Configure the security checks for include files. This statement takes a list of arguments, separated by white space. The following arguments are recognized:

all   Enable all checks.

owner   The file is not owned by root.

iwgrp
groupwritablefile
               The file is group writable.

iwoth
worldwritablefile
               The file is world writable.

dir_iwgrp
groupwritabledir
               The file resides in a group writable directory.

dir_iwoth
worldwritabledir
               The file resides in a world writable directory.

link   The file is a symbolic link to a file residing in a group or world
               writable directory.

Each of the above keywords may be prefixed by 'no', which reverses its
meaning. The special keyword 'none' is synonymous to 'noall', i.e. it
disables all checks. Each keyword adds or removes a particular test to the
existing check list, which is initialized as described in [security checks],
page 7. Thus, the foll owning statement results in all checks, except for
the file ownership:

        include-security noowner

In the example below, the check list is first cleared by using the noall
statement, and then a set of checks is added to it:

        include-security noall owner iwoth iwgrp

The include-security statement is global, i.e. it affects all include
statements appearing below it, up to the next include-security state-
ment, or end of configuration file, whichever occurs first.

    The include statement can appear in any place of the configuration file,
both within or outside a rule.

    This statement provides a convenient way for user-dependent rush con-
figuration. For example, the following fall-through rule (see Section 4.2.5
[Fall-through], page 18) allows to keep each user's configuration in a file
named .rush, located in the user's home directory:

     rule user
        inlcude ~/.rush
        fall-through

Of course, it is supposed that such a per-user file, if it exists, is writable
only for super-user and does not contain any rule statements.

    The use of include files may be especially useful for per-user localization
(see Section 4.2.10 [Localization], page 22). It suffices to provide a fall-

through rule, similar to the one above, and to place a `locale` directive in
`~/.rush` files, according to the users' preferences.

4.4 Debugging

The `debug` statement sets the *debugging level* – an integer value that controls
the verbosity of `rush`:

`debug num`   [Configuration]
    Set the debugging level to *num*.

    The greater *num* is, the more verbose is the logging. The debugging
information is reported via `syslog` at facility '`authpriv`', priority '`debug`'.
As of version 1.8, the following debugging levels are supported:

1   A minimum debugging level, and the only one whose messages
          are logged using the priority '`notice`'. At this level, `rush` only
          logs requests and rules selected to handle them. For example:

```
rush[16821]: Serving request "/usr/libexec/sftp-server"
for sergiusz by rule sftp-savane
```

2   List all actions executed when serving requests.

3   Verbosely describe parsing of the configuration file.

    More debugging levels may be implemented in future.

4.5 Regex

The `regex` statement configures the flavor of regular expressions for use by
subsequent `command`, `match` and `transform` statements.

`regex regex-flags`   [Configuration]
    Configure the type of regular expressions.

    Regex-flags is a whitespace-separated list of flags. Each flag is a word
specifying some regular expression feature. It can be preceded by '`+`' to
enable this feature (this is the default), or by '`-`' to disable it. Valid flags
are:

'`extended`'
          Use POSIX Extended Regular Expression syntax when interpret-
          ing regex. This is the default.

'`basic`'   Use basic regular expressions. Equivalent to '`-extended`'.

'`icase`'   Do not differentiate case. Subsequent regex matches will be case
          insensitive.

    For example, the following statement enables POSIX extended, case in-
sensitive matching:

```
regex +extended +icase
```

The `regex` settings affect subsequent `command` and `match` statements (see Section 4.2.1 [Conditions], page 9), and remain in effect until next `regex` statement or the end of configuration file, whichever occurs first.

4.6 Sleep Time

`sleep-time` *number*   [Configuration]
> Set the time in seconds to sleep before exiting on error.

This statement is intended as a measure against brute-force attacks. Default sleep time is 5 seconds.

4.7 Error Messages

A set of statements configures textual messages which GNU Rush returns to the user if an error of some class occurs. All of them take a single argument which is subject to backslash interpretation, as described in Section 4.2.8 [Exit], page 20.

`usage-error` *text*   [Configuration]
> Define a textual message which is returned to the remote party if a usage error occurs.
>
> Default is:
>
> ```
> You are not permitted to execute this command.
> ```

`nologin-error` *text*   [Configuration]
> Define a textual message which is returned to the remote user if there is no such user name in the password database.
>
> Default is:
>
> ```
> You do not have interactive login access to this machine.
> ```

`config-error` *text*   [Configuration]
> Define a textual message which is returned to the remote party on Rush configuration errors.
>
> Default is:
>
> ```
> Local configuration error occurred.
> ```

`system-error` *text*   [Configuration]
> Define a textual message which is returned to the remote party if a system error occurs.
>
> Default message is:
>
> ```
> A system error occurred while attempting to execute command.
> ```

5 Default Configuration

You can compile `rush` with the default configuration built in the binary. Such a binary can then be run without configuration file. However, if a configuration file is present, it takes priority over the built-in configuration.

To compile `rush` with the built-in configuration, first compile the package as usual. Then, prepare a configuration file, and test it using `rush --lint`. If the test shows no errors, reconfigure the package, using the `--with-default-config` option:

```
./configure --with-default-config=file
```

where *file* is the name of your configuration file. Then, recompile and install the package.

You can inspect the built-in configuration using the `--show-default` option:

```
rush --show-default
```

6 Usage Tips

In this chapter we will explain how to write GNU Rush configuration rules for several popular remote copy and version control system utilities. For this purpose, we assume the following setup:

- Users are allowed to use `scp` and `rsync` to upload files to the `/incoming` directory and to copy files to and from their `~/public_html` directory. The `/incoming` directory is located on server in `/home/ftp` directory, but that is transparent to users, i.e. they use `scp file host:/incoming` (not `host:/home/ftp/incoming`) to upload files.

- Additionally, users may use `sftp` to manage their `~/public_html` directory. In this case, to prevent users from accessing other directories, `sftp-server` is executed in a chrooted environment.

- The server runs three version control system repositories, whose corresponding repositories are located in the following directories:

VCS	Repository Root
cvs	/cvsroot
svn	/svnroot
git	/gitroot

6.1 scp

The `scp` utility is executed on the server side with option `-t`, when copying files to server, and with `-f` when copying from it. Thus, the basic templates for `scp` rules are:

```
# Copying to server:
rule scp-to
  command ^scp -t
  ...

# Copying from server:
rule scp-from
  command ^scp -f
  ...
```

You may also wish to allow for `-v` ('verbose') command line option. In this case, the 'scp-to' rule will become:

```
rule scp-to
  command ^scp (-v )?-t
  ...
```

First, we want users to be able to upload files to `/home/ftp/incoming` directory. Moreover, the `/home/ftp` directory prefix must be invisible to them. We must also make sure that the user cannot get outside the `incoming` directory by using `../` components in his upload path. So, our first rule for `scp` uploads will be:

```
rule scp-to-incoming
```

```
command ^scp (-v )?-t /incoming/
match[$] ! \.\./
set[0] /bin/scp
transform[$] s|^|/home/ftp/|
```

The `match[$]` statement ensures that no relative components are used. Two transform rules ensure that the right `scp` binary is used and that `/home/ftp` prefix is prepended to the upload path.

Other than uploading to `/incoming`, users must be able to use `scp` to manage `public_html` directories located in their homes. They should use relative paths for that, i.e., the command:

```
$ scp file.html server:
```

will copy file `file.html` to `~/public_html/file.html` on the server. The corresponding rule is:

```
rule scp-home
  command ^scp (-v )?-[tf] [^/].*
  match[$] ! \.\./
  set[0] /bin/scp
  transform[$] s|^|public_html/|
  chdir ~
```

Finally, we provide two trap rules for diagnostic purposes:

```
rule scp-to-trap
  command ^scp (-v )?-t
  exit Error: Uploads to this directory prohibited

rule scp-from
  command ^scp (-v )?-f
  exit Error: Downloads from this directory prohibited
```

6.2 rsync

On the server side, `rsync` is executed with the `--server` command line option. In addition, when copying files from the server, the `--sender` option is used. This allows to discern between incoming and outgoing requests.

In our setup, `rsync` is used the same way as `scp`, so the two rules will be:

```
rule rsync-incoming
  command ^rsync --server
  command ! --sender
  match[$] /incoming/
  match[$] ! \.\./
  transform[0] s|^|/usr/bin/|
  transform[$] s|^|/home/ftp/|

rule rsync-home
  command ^rsync
  match[$] ! ^[^/]
  match[$] ! \.\./
  transform[0] s|^|/usr/bin/|
  transform[$] s|^|public_html/|
```

```
      chdir ~
```
The trap rules for `rsync` are trivial:
```
 rule rsync-to-trap
   command ^rsync
   command --sender
   exit Error: Downloads from this directory prohibited

 rule rsync-from-trap
   command ^rsync
   exit Error: Uploads to this directory prohibited
```

6.3 sftp

Executing `sftp` on the client machine invokes `sftp-server`, without arguments, on the server.

We want to allow our users to use `sftp` to manage their `public_html` directories. The `sftp-server` will be executed with the user's home directory as root, in a chrooted environment. For this to work, each user's home must contain a copy of `sftp-server` (which we'll place in `~/bin` subdirectory) and all files it needs for normal execution: `/etc/group` and `/etc/passwd` with one entry (for the user and his group), and, unless the binary is linked statically, all the shared libraries it is linked with, in the subdirectory `~/lib`.

Given these prerequisites, the following rule will ensure proper `sftp` interaction:
```
    rule sftp-incoming
      command ^.*/sftp-server
      set[0] /bin/sftp-server
      chroot ~
      chdir public_html
```
Notice the last action. Due to it, users don't have to type `cd public_html` at the beginning of their sftp sessions.

6.4 cvs

Using `cvs` over `ssh` invokes `cvs server` on the server machine. In the simplest case, the following rule will do to give users access to CVS repositories:
```
    rule cvs
      command ^cvs server
      transform s|^cvs|/usr/bin/cvs -f
```
However, `cvs` as of version 1.12.13 does not allow to limit root directories that users are allowed to access. It does have `--allow-root` option, but unfortunately this option is ignored when invoked as `cvs server`. To restrict possible roots, we have to run `cvs` in a chrooted environment. Let's suppose we created an environment for `cvs` in directory `/var/cvs`, with the `cvs` binary located in `/var/cvs/bin` and repository root directory being `/var/cvs/cvsroot`. Then, we can use the following rule:
```
    rule cvs
```

```
command ^cvs server
set[0] /bin/cvs
chroot /var/cvs
```

6.5 svn

Remote access to SVN repositories is done via `svnserve` binary. It is executed on server with `-t` option. The `-r` option can be used to restrict access to a subset of root directories. So, we can use the following rule:

```
rule svn
  command ^svnserve -t
  transform s|-r *[^ ]*||;s|^svnserve |/usr/bin/svnserve -r /svnroot|
```

The `transform` action removes any `-r` options the user might have specified and enforces a single root directory. A more restrictive action can be used to improve security:

```
transform s|.*|/usr/bin/svnserve -r /svnroot|
```

6.6 git

Remote access to Git repositories over ssh causes execution of `git-receive-pack` and `git-upload-pack` on the server. The simplest rule for Git is:

```
rule git
  command ^git-(receive|upload)-pack
  transform[0] s|^|/usr/bin/|
```

The `transform` action is necessary to ensure the proper location of Git binaries to use. This example supposes they are placed in `/usr/bin`, you will have to tailor it if they are located elsewhere on your system.

To limit Git accesses to repositories under `/gitroot` directory, use `match[1]` construct, as shown in the example below:

```
rule git
  command ^git-(receive|upload)-pack
  match[1] ^/gitroot[^ ]+\.git$
  transform[0] s|^|/usr/bin/|
```

To provide more helpful error messages, you may follow this rule by a trap rule (see Section 4.2.8 [Exit], page 20):

```
# Trap the rest of Git requests:
rule git-trap
  command ^git-.+
  exit fatal: access to this repository is denied.
```

6.7 Notification

In this section we will show how to set up a mail notification for Rush rules. Let's suppose we wish to receive emails for each upload by `scp-to` rule (see Section 6.1 [scp], page 31). To do so, we add the following fall through rule to the beginning of `rush.rc`:

```
rule default
```

```
    post-socket inet://localhost
    fall-trough
```

This will enable notifications for each rule located below this one. Missing port in `post-socket` statement means `rush` will be using the default 'tcpmux' port.

To receive and process these requests, you will need an `inetd` capable to handle TCPMUX. We recommend the one from GNU Inetutils package (GNU Inetutils (`http://www.gnu.org/software/inetutils`)). In /etc/inetd.conf file, we add:

```
# Enable TCPMUX handling.
tcpmux           stream  tcp  nowait root  internal
# Handle 'scp-to' service.
tcpmux/+scp-to  stream  tcp  nowait root  /usr/sbin/tcpd  /bin/rushmail
```

The program /bin/rushmail does the actual notification. Following is its simplest implementation:

```
#! /bin/sh

read user command

/usr/sbin/sendmail -oi -t <<EOT
From: GNU Rush Notification <devnull@localhost>
To: <root@localhost>
Subject: GNU Rush notification

Be informed that $user executed $command.
EOT
```

7 Test Mode

GNU Rush provides a special *test mode*, intended to test configuration files and to emulate execution of commands. The test mode is enabled by `--test` command line option (aliases: `--lint`, `-t`). When `rush` is given this option, the following occurs:

1. All diagnostic messages are redirected to standard error, instead of syslog.

2. If a single non-option argument is present, it is taken as a name of the configuration file to use.

3. The configuration file is parsed. If parsing fails, the program exits with the code 1.

4. If the `-c` option is present, `rush` processes its argument as usual (see Chapter 2 [Operation], page 3), except that the command itself is not executed.

5. Otherwise, if `-i` option is present, `rush` emulates interactive usage, but does not execute the final command.

An exit status of 0 means no errors, 1 means an error has occurred.

You may also emulate access by a particular user, by supplying his user name via the `--user` (`-u`) option. This option implies `--test`.

In test mode, you may set debugging level (see Section 4.4 [Debugging], page 27) from the command line, using the `--debug` (`-d`) command line option. It expects a single number specifying debugging level as its argument. The debugging level set this way overrides settings from the configuration file.

Following are several examples that illustrate the use of test mode in various cases:

1. Test default configuration file:
   ```
   $ rush --test
   ```
2. Test configuration file `sample.rc`:
   ```
   $ rush --test sample.rc
   ```
3. Test interactive access
   ```
   $ rush --test -i sample.rc
   ```
4. Test the configuration file and emulate execution of the command `cvs server`. Use debugging level 2:
   ```
   $ rush --test --debug=2 -c "cvs server"
   ```
5. Same, but for user 'jeff':
   ```
   $ rush --user=jeff --debug=2 -c "cvs server"
   ```
   Note, that you don't need to specify `--test` along with `--user` or `-i` options.
6. Same, but use `sample.rc` instead of the default configuration file:
   ```
   $ rush --test --debug=2 -c "cvs server" sample.rc
   ```

7.1 Dump Mode

Dump mode is similar to test mode. The main difference is that in this mode, `rush` dumps on the standard error a description of the user request after performing all checks and transformations.

The mode is requested by the `--dump=attr` (`-D attr`) option. The argument *attr* is a comma-separated list of the names of attributes to be included in the dump, or the word '`all`', standing for all attributes.

Additional options and arguments are the same as for the `--test` option.

The description is formatted as a JSON object[1] with the following attributes. These are also the allowed values for the *attr* list:

cmdline Command line after transformations.

argv   Array of command line arguments after transformations.

prog   Name of the program to be executed. If '`null`', `argv[0]` will be used.

interactive '`0`' for normal requests, '`1`' for interactive requests.

pw_name Name of the user from the system user database.

pw_uid UID of the user.

pw_gid GID of the user.

pw_dir Home directory of the user, as set in the system user database.

umask Value of the umask (octal).

chroot_dir Chroot directory.

home_dir Current working directory.

gid   New GID as set by the `newgrp` action, or '`-1`' if unchanged.

fork   Fork mode. It is a three-state attribute: '`0`' meaning *disabled*, '`1`' meaning *enabled*, and '`-1`' meaning *default state*.

acct   Accounting mode. See '`fork`', for a description of possible values.

text_domain
     Textual domain for i18n.

localedir Locale directory for i18n.

locale   Locale name

environ Dump of the environment (array of assignments).

The attribute '`all`' stands for all attribute in the same order as listed in the table above.

---

[1] Well, almost. It diverges from the JSON standard in that slash characters are not escaped in string objects.

8 Option Summary

This chapter provides a short summary of `rush` command line options.

`-c command`
> Specify the command to run.

`-C test`
`--security-check=test`
> Configure security checks for the main configuration file. See [include-security], page 25, for the description of *test* argument. See [security checks], page 7, for the discussion of the available security tests.

`-d number`
`--debug=number`
> Set debugging level.

`--dump=attrs`
`-D attrs`   Run in *request dump mode*. Argument is a comma-separated list of attribute names. See Section 7.1 [dump mode], page 38, for a detailed description of the request dump mode.

`-i`   Emulate interactive access. See Chapter 7 [Test Mode], page 37.

`--show-default`
> Display the default built-in configuration. See Chapter 5 [Default Configuration], page 29, for more information.

`-t`
`--test`
`--lint`   Run in test mode. An optional argument may be used with this option to specify alternative configuration file name, e.g.:

```
$ rush --lint ./test.rc
```

> If the `-c` option is also specified, `rush` emulates the normal processing for the command, but does not execute it.

`-u name`
`--user=name`
> Emulate access by user *name*. This option implies `--test` and is valid only when used by root and in conjunction with the `-c` option.

`-v`
`--version`
> Display program version.

`-h`
`--help`   Display a short help message.

`--usage`   Display a concise usage summary.

9 The `rushwho` utility.

The `rushwho` utility displays a list of users who are currently using `rush`. The utility operates on default Rush database, which is maintained if `rush` runs in accounting mode (see Section 4.2.6 [Accounting and Forked Mode], page 19). The following is a sample output from `rushwho`:

```
Login      Rule     Start     Time       PID      Command
jeff       sftp     Sun 12:17 00:58:26   10673    bin/sftp-server
```

The information displayed is:

Login   The login name of the user.

Rule   The tag of the rule he is served under (see Section 4.2 [Rule], page 9).

Start   Time when the rule began execution.

Time   Duration of the session.

PID   PID of the running command.

Command   Command line being executed.

This format is a built-in default. It may be changed either by setting the `RUSHWHO_FORMAT` environment variable to the desired format string, or by using `--format` command line option.

9.1 Rushwho Options

This section summarizes the command line options understood by `rushwho` utility.

`-F string`
`--format=string`

Use *string* instead of the default format, described in Chapter 9 [Rushwho], page 41. See Section 9.2 [Formats], page 42, for a detailed description of the output format syntax. If *string* begins with a '`@`', then this character is removed from it, and the resulting string is regarded as a name of the file to read. The contents of this file is the format string. The file is read literally, except that lines beginning with '`;`' are ignored (they can be used to introduce comments). For example, `rushwho --format=@formfile` reads in the contents of the file named `formfile`.

`-f dir`
`--file=dir`

Use database directory *dir*, instead of the default. By default, database files are located in `/usr/local/var/rush`.

```
-H
--no-header
```
Do not display header line.

```
-v
--version
```
Display program version.

```
-h
--help
```
Display a short help message.

`--usage`   Display a concise usage summary.

9.2 Output Formats

A format string controls the output of every record from GNU Rush accounting database. It may contain following four types of objects:

Ordinary characters
These are copied to the output verbatim.

Escapes   An escape is a backslash ('\\'), followed by a single character. It is interpreted as follows:

Escape	Output
\a	Audible bell character (ASCII 7)
\b	Backspace character (ASCII 8)
\e	Escape character (ASCII 27)
\f	Form-feed character (ASCII 12)
\n	Newline character (ASCII 10)
\r	Carriage return character (ASCII 13)
\t	Horizontal tabulation character (ASCII 9)
\v	Vertical tabulation character (ASCII 11)
\\	A single backslash ('\')
\"	A double-quote.

Any escape not listed in the table above results in its second character being output.

Quoted strings
Strings are delimited by single or double quotes. Within a string any escape sequences are interpreted as described above.

Format specifications
A *format specification* is a kind of function, which outputs a particular piece of information from the database record.

   Each format specification starts with an opening brace and ends with a closing brace. The first word after the brace is the name of the format

specification. The rest of words are *positional arguments* followed by *keyword arguments*. Both are optional. When specified, keyword arguments must follow positional ones. A keyword argument begins with a colon. For example:

(time)   A single format specification.

(time 10)   The same format specification with the output width limited to 10 characters.

(time 10 Duration)
        The '`time`' format specification, with the output width limited to 10 characters and '`Duration`' as a header title.

(time 10 "Session Duration" :right :format %H:%M)
        The same with two keyword arguments: '`:right`' and '`:format`'. The latter takes the string '`%H:%M`' as its argument. Notice the use of quoted string to preserve the whitespace.

    The full list of format specifications follows.

`newline` [*count*]   [Format Spec]
    Causes the newline character to be output. If the optional *count* is supplied, that many newlines will be printed

`tab` [*num*]   [Format Spec]
    Advance to the next tab stop in the output stream. If optional *num* is present, then skip *num* tab stops. Each tab stop is eight characters long.

    The following specifications output particular fields of a database record. They all take two positional arguments: *width* and *title*.

    The first argument, *width* sets the maximum output length for this specification. If the number of characters actually output is less than the width, they will be padded with whitespace either to the left or to the right, depending on the presence of the `:right` keyword argument. If the number of characters is greater than *width*, they will be truncated to fit. If *width* is not given, the exact data are output as is.

    The second argument, *title*, gives the title of this column for the heading line. By default no title is output.

    Every field specification accepts at least two keyword arguments. The keyword `:right` may be used to request alignment to the right for the data. This keyword is ignored if *width* is not given.

    The keyword `:empty` followed by a string causes `rushwho` to output that string if the resulting value for this specification would otherwise be empty.

`user` *width* *title* [:*empty* *repl*][:*right*]   [Format Spec]
    Print the user login name.

`time` *`width title`* [:*empty* `repl`][:*right*][:*format*   [Format Spec]
       `date-format`]
`start-time` *`width title`* [:*empty*   [Format Spec]
       `repl`][:*right*][:*format* `date-format`]
   Date and time when the session started.

   The :`format` keyword introduces the `strftime` format string to be used
   when converting the date for printing. The default value is '`%a %H:%M`'.
   See Appendix A [Time and Date Formats], page 51, for a detailed de-
   scription of `strftime` format strings.

`stop-time` *`width title`* [:*empty*   [Format Spec]
       `repl`][:*right*][:*format* `date-format`]
   Time when the command finished. This specifier has sense only for
   `rushlast` (see Chapter 10 [Rushlast], page 45). If the command is still
   running, the word '`running`' is output.

`duration` *`width title`* [:*empty* `repl`][:*right*]   [Format Spec]
   Total time of the session duration.

`rule` *`width title`* [:*right*]   [Format Spec]
   The tag of the rule used to serve the user. See Section 4.2 [Rule], page 9,
   for a detailed description of rules and tags.

`command` *`width title`* [:*empty* `repl`][:*right*]   [Format Spec]
   Command line being executed.

`pid` *`width title`* [:*right*]   [Format Spec]
   PID of the process.

   For example, the following is the default format for the `rushwho` util-
ity. It is written in a form, suitable for use in a file supplied with the
`--format=@file` command line option (see [format option], page 41):

```
(user 10 Login)" "
(rule 8 Rule)" "
(start-time 0 Start)" "
(duration 9 Time)" "
(pid 10 PID)" "
(command 28 Command)
```

10 The `rushlast` utility.

The `rushlast` utility searches back through the GNU Rush database and displays a list of all user sessions since the database was created. By default, it displays the following information:

```
Login      Rule     Start      Stop     Time    Command
sergiusz   rsync    Sun 20:43 Sun 20:43 05:57   /usr/bin/rsync /upload
jeff       sftp-sav Sun 20:09 running   07:17   /bin/sftp-server
```

Login   The login name of the user.

Rule   The tag of the rule he is served under (see Section 4.2 [Rule], page 9).

Start   Time when the rule began execution.

Start   Time when the command finished, or the word 'running' if it is still running.

Time   Duration of the session.

Command   Command line being executed.

This format is a built-in default. It may be changed either by setting the `RUSHLAST_FORMAT` environment variable to the desired format string, or by using `--format` command line option (see Section 10.1 [Rushlast Options], page 45).

10.1 Rushlast Options

This section summarizes the command line options understood by `rushlast` utility.

`-F string`
`--format=string`

> Use *string* instead of the default format, described in Chapter 9 [Rushwho], page 41. See Section 9.2 [Formats], page 42, for a detailed description of the output format syntax. If *string* begins with a '@', then this character is removed from it, and the resulting string is regarded as a name of a file to read. The contents of this file is the format string. The file is read literally, except that lines beginning with ';' are ignored (they can be used to introduce comments). For example, `rushwho --format=@formfile` reads in the contents of the file named `formfile`.

`-f dir`
`--file=dir`

> Use database directory *dir*, instead of the default. By default, database files are located in `/usr/local/var/rush`.

`--forward`

> Display entries in chronological order, instead of the reverse chronological one, which is the default.

`-n `*`number`*
`--count=`*`number`*
`-`*`number`* Show at most *number* records. The form `-`*`number`* is provided for compatibility with the *last(1)* utility.

`-H`
`--no-header`
Do not display header line.

`-v`
`--version`
Display program version.

`-h`
`--help` Display a short help message.

`--usage` Display a concise usage summary.

11 Accounting Database

Rush accounting database is stored in the directory *localstatedir*/rush, where *localstatedir* stands for the name of the local state directory, defined at compile time. By default, it is *prefix*/var, where *prefix* is the installation prefix, which defaults to /usr/local. Thus, the default database directory is /usr/local/var/rush. You can change this default by using --localstatedir option to configure before compiling the package. The --prefix option affects it as well.

As of version 1.8, the database consists of two files, called utmp and wtmp. The wtmp file keeps information about all user sessions, both finished and still active. The utmp file contains indices to those records in wtmp, which represent active sessions.

The wtmp grows continuously, while utmp normally grows the first day or two after enabling accounting mode, and from then on its size remains without changes. If you set up log file rotation, e.g. by using logrotate (see Section "logrotate" in *logrotate man page*), or a similar tool, it is safe to rotate wtmp without notifying rush. The only requirement is to truncate utmp to zero size after rotating wtmp, as shown in the following logrotate.conf snippet:

```
/var/run/rush/wtmp {
    monthly
    create 0640 root svusers
    postrotate
      cat /dev/null > /var/run/rush/utmp
    endscript
}
```

Accounting files are owned by 'root' and normally have permissions '600'. You may change the default permissions using the following configuration file statements:

acct-umask *mask* [Rule Config]
  Set umask used when accessing accounting database files. Default value is '022'.

acct-dir-mode *mode* [Rule Config]
  Set mode bits for the accounting directory. The *mode* argument is the mode in octal.

acct-file-mode *mode* [Rule Config]
  Set mode bits for wtmp and utmp files.

Notice, that these statements affect file and directory modes only when the corresponding file or directory is created. Rush will not change modes of the existing files.

The following sections contain a detailed description of the structure of these two files. You may skip them, if you are not interested in technical details.

11.1 The `wtmp` file

The `wtmp` file consists of variable-size entries. It is designed so that it can easily be read in both directions.

Each record begins with a fixed-size header, which is followed by three zero-terminated strings, and the record size in `size_t` representation. The three strings are, in that order: the user login name, the rule tag, and the full command line.

The header has the following structure:

```
struct rush_wtmp {
        size_t reclen;
        pid_t pid;
        struct timeval start;
        struct timeval stop;
        char *unused[3];
};
```

where:

reclen   is the length of the entire record, including the size of this header. This field is duplicated at the end of the record.

pid   is the PID of the command executed for the user.

start   represents the time of the beginning of the user session.

stop   represents the time when the user session finished. If the session is still running, this field is filled with zeros.

unused   The three pointers at the end of the structure are used internally by `rush`. On disk, these fields are always filled with zeros.

11.2 The `utmp` file

The `utmp` file consists of a fixed-size records of the following structure:

```
struct rush_utmp {
        int status;
        off_t offset;
};
```

The fields have the following meaning:

status   Status of the record: '0' if the record is unused, and '1' if it represents an active session.

offset   Offset to the corresponding record in `wtmp` (see previous section).

12 How to Report a Bug

Email bug reports to `bug-rush@gnu.org`. Please include a detailed description of the bug and information about the conditions under which it occurs, so we can reproduce it. To facilitate the task, the following list shows the basic set of information that is needed in order to find the bug:

- Package version you use.
- A detailed description of the bug.
- Conditions under which the bug appears.
- It is often helpful to send the contents of `config.log` file along with your bug report. This file is created after running `./configure` in the GNU Rush source root directory.

Appendix A Time and Date Formats

This appendix documents the time format specifications understood by the `:format` keyword in time output format specifiers (see Section 9.2 [Formats], page 42). Essentially, it is a reproduction of the man page for GNU `strftime` function.

Ordinary characters placed in the format string are reproduced without conversion. Conversion specifiers are introduced by a '`%`' character, and are replaced as follows:

%a   The abbreviated weekday name according to the current locale.

%A   The full weekday name according to the current locale.

%b   The abbreviated month name according to the current locale.

%B   The full month name according to the current locale.

%c   The preferred date and time representation for the current locale.

%C   The century number (year/100) as a 2-digit integer.

%d   The day of the month as a decimal number (range 01 to 31).

%D   Equivalent to '`%m/%d/%y`'.

%e   Like '`%d`', the day of the month as a decimal number, but a leading zero is replaced by a space.

%E   Modifier: use alternative format, see below (see [conversion specs], page 53).

%F   Equivalent to '`%Y-%m-%d`' (the ISO 8601 date format).

%G   The ISO 8601 year with century as a decimal number. The 4-digit year corresponding to the ISO week number (see '`%V`'). This has the same format and value as '`%y`', except that if the ISO week number belongs to the previous or next year, that year is used instead.

%g   Like '%G', but without century, i.e., with a 2-digit
                year (00-99).

%h   Equivalent to '%b'.

%H   The hour as a decimal number using a 24-hour
                clock (range 00 to 23).

%I   The hour as a decimal number using a 12-hour
                clock (range 01 to 12).

%j   The day of the year as a decimal number (range
                001 to 366).

%k   The hour (24-hour clock) as a decimal number
                (range 0 to 23); single digits are preceded by a
                blank. (See also '%H'.)

%l   The hour (12-hour clock) as a decimal number
                (range 1 to 12); single digits are preceded by a
                blank. (See also '%I'.)

%m   The month as a decimal number (range 01 to 12).

%M   The minute as a decimal number (range 00 to 59).

%n   A newline character.

%O   Modifier: use alternative format, see below (see
                [conversion specs], page 53).

%p   Either 'AM' or 'PM' according to the given time
                value, or the corresponding strings for the current
                locale. Noon is treated as 'pm' and midnight as 'am'.

%P   Like '%p' but in lowercase: 'am' or 'pm' or a corre-
                sponding string for the current locale.

%r   The time in 'a.m.' or 'p.m.' notation. In the
                POSIX locale this is equivalent to '%I:%M:%S %p'.

%R   The time in 24-hour notation ('%H:%M'). For a ver-
                sion including the seconds, see '%T' below.

%s   The number of seconds since the Epoch, i.e., since
                1970-01-01 00:00:00 UTC.

%S	The second as a decimal number (range 00 to 61).
%t	A tab character.
%T	The time in 24-hour notation (`%H:%M:%S`).
%u	The day of the week as a decimal, range 1 to 7, Monday being 1. See also `%w`.
%U	The week number of the current year as a decimal number, range 00 to 53, starting with the first Sunday as the first day of week 01. See also `%V` and `%W`.
%V	The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week. See also `%U` and `%W`.
%w	The day of the week as a decimal, range 0 to 6, Sunday being 0. See also `%u`.
%W	The week number of the current year as a decimal number, range 00 to 53, starting with the first Monday as the first day of week 01.
%x	The preferred date representation for the current locale without the time.
%X	The preferred time representation for the current locale without the date.
%y	The year as a decimal number without a century (range 00 to 99).
%Y	The year as a decimal number including the century.
%z	The time-zone as hour offset from GMT. Required to emit RFC822-conformant dates (using `%a, %d %b %Y %H:%M:%S %z`)
%Z	The time zone or name or abbreviation.
%+	The date and time in *date(1)* format.
%%	A literal `%` character.

Some conversion specifiers can be modified by preceding them by the 'E' or 'O' modifier to indicate that an alternative format should be used. If the alternative format or specification does not exist for the current locale, the behaviour will be as if the unmodified conversion specification were used. The Single Unix Specification mentions '%Ec', '%EC', '%Ex', '%EX', '%Ry', '%EY', '%Od', '%Oe', '%OH', '%OI', '%Om', '%OM', '%OS', '%Ou', '%OU', '%OV', '%Ow', '%OW', '%Oy', where the effect of the 'O' modifier is to use alternative numeric symbols (say, roman numerals), and that of the 'E' modifier is to use a locale-dependent alternative representation.

Appendix B GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

`http://fsf.org/`

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or

to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties— for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not

add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new

versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ''GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Concept Index

This is a general index of all issues discussed in this manual.