NAME

rsyncd.conf — configuration file for rsync in daemon mode

SYNOPSIS

rsyncd.conf

DESCRIPTION

The rsyncd.conf file is the runtime configuration file for rsync when run as an rsync daemon.

The rsyncd.conf file controls authentication, access, logging and available modules.

FILE FORMAT

The file consists of modules and parameters. A module begins with the name of the module in square brackets and continues until the next module begins. Modules contain parameters of the form “name = value”.

The file is line-based — that is, each newline-terminated line represents either a comment, a module name or a parameter.

Only the first equals sign in a parameter is significant. Whitespace before or after the first equals sign is discarded. Leading, trailing and internal whitespace in module and parameter names is irrelevant. Leading and trailing whitespace in a parameter value is discarded. Internal whitespace within a parameter value is retained verbatim.

Any line beginning with a hash (#) is ignored, as are lines containing only whitespace.

Any line ending in a \ is “continued” on the next line in the customary UNIX fashion.

The values following the equals sign in parameters are all either a string (no quotes needed) or a boolean, which may be given as yes/no, 0/1 or true/false. Case is not significant in boolean values, but is preserved in string values.

LAUNCHING THE RSYNC DAEMON

The rsync daemon is launched by specifying the **−−daemon** option to rsync.

The daemon must run with root privileges if you wish to use chroot, to bind to a port numbered under 1024 (as is the default 873), or to set file ownership. Otherwise, it must just have permission to read and write the appropriate data, log, and lock files.

You can launch it either via inetd, as a stand-alone daemon, or from an rsync client via a remote shell. If run as a stand-alone daemon then just run the command “**rsync −−daemon**” from a suitable startup script.

When run via inetd you should add a line like this to /etc/services:

```
rsync           873/tcp
```

and a single line something like this to /etc/inetd.conf:

```
rsync   stream  tcp     nowait  root   /usr/bin/rsync rsyncd −−daemon
```

Replace “/usr/bin/rsync” with the path to where you have rsync installed on your system. You will then need to send inetd a HUP signal to tell it to reread its config file.

Note that you should **not** send the rsync daemon a HUP signal to force it to reread the `rsyncd.conf` file. The file is re-read on each client connection.

GLOBAL PARAMETERS

The first parameters in the file (before a [module] header) are the global parameters.

You may also include any module parameters in the global part of the config file in which case the supplied value will override the default for that parameter.

motd file
This parameter allows you to specify a “message of the day” to display to clients on each connect. This usually contains site information and any legal notices. The default is no motd file.

pid file This parameter tells the rsync daemon to write its process ID to that file. If the file already exists, the rsync daemon will abort rather than overwrite the file.

port You can override the default port the daemon will listen on by specifying this value (defaults to 873). This is ignored if the daemon is being run by inetd, and is superseded by the **−−port** command-line option.

address
You can override the default IP address the daemon will listen on by specifying this value. This is ignored if the daemon is being run by inetd, and is superseded by the **−−address** command-line option.

socket options
This parameter can provide endless fun for people who like to tune their systems to the utmost degree. You can set all sorts of socket options which may make transfers faster (or slower!). Read the man page for the `setsockopt()` system call for details on some of the options you may be able to set. By default no special socket options are set. These settings can also be specified via the **−−sockopts** command-line option.

MODULE PARAMETERS

After the global parameters you should define a number of modules, each module exports a directory tree as a symbolic name. Modules are exported by specifying a module name in square brackets [module] followed by the parameters for that module. The module name cannot contain a slash or a closing square bracket. If the name contains whitespace, each internal sequence of whitespace will be changed into a single space, while leading or trailing whitespace will be discarded.

comment
This parameter specifies a description string that is displayed next to the module name when clients obtain a list of available modules. The default is no comment.

path This parameter specifies the directory in the daemon’s filesystem to make available in this module. You must specify this parameter for each module in `rsyncd.conf`.

use chroot
If “use chroot” is true, the rsync daemon will chroot to the “path” before starting the file transfer with the client. This has the advantage of extra protection against possible implementation security holes, but it has the disadvantages of requiring super-user privileges, of not being able to follow symbolic links that are either absolute or outside of the new root path, and of complicating the preservation of users and groups by name (see below).

As an additional safety feature, you can specify a dot-dir in the module’s “path” to indicate the point where the chroot should occur. This allows rsync to run in a chroot with a non−"/” path for the top of the transfer hierarchy. Doing this guards against unintended library loading (since those absolute paths will not be inside the transfer hierarchy unless you have used an unwise pathname), and lets you setup libraries for the chroot that are outside of the transfer. For example, specifying “/var/rsync/./module1” will chroot to the “/var/rsync” directory and set the inside-chroot path to “/module1”. If you had omitted the dot-dir, the chroot would have used the whole path, and the inside-chroot path would have been “/”.

When “use chroot” is false or the inside-chroot path is not “/”, rsync will: (1) munge symlinks by default for security reasons (see “munge symlinks” for a way to turn this off, but only if you trust your users), (2) substitute leading slashes in absolute paths with the module’s path (so that options such as **−−backup−dir**, **−−compare−dest**, etc. interpret an absolute path as rooted in the module’s "path" dir), and (3) trim ".." path elements from args if rsync believes they would escape the module hierarchy. The default for "use chroot" is true, and is the safer choice (especially if the module is not read-only).

When this parameter is enabled, rsync will not attempt to map users and groups by name (by default), but instead copy IDs as though **−−numeric−ids** had been specified. In order to enable name-mapping, rsync needs to be able to use the standard library functions for looking up names and IDs (i.e. `getpwuid()` , `getgrgid()` , `getpwname()` , and `getgrnam()`). This means the rsync process in the chroot hierarchy will need to have access to the resources used by these library functions (traditionally /etc/passwd and /etc/group, but perhaps additional dynamic libraries as well).

If you copy the necessary resources into the module's chroot area, you should protect them through your OS's normal user/group or ACL settings (to prevent the rsync module's user from being able to change them), and then hide them from the user's view via "exclude" (see how in the discussion of that parameter). At that point it will be safe to enable the mapping of users and groups by name using the "numeric ids" daemon parameter (see below).

Note also that you are free to setup custom user/group information in the chroot area that is different from your normal system. For example, you could abbreviate the list of users and groups.

numeric ids

Enabling this parameter disables the mapping of users and groups by name for the current daemon module. This prevents the daemon from trying to load any user/group-related files or libraries. This enabling makes the transfer behave as if the client had passed the **−−numeric−ids** command-line option. By default, this parameter is enabled for chroot modules and disabled for non-chroot modules.

A chroot-enabled module should not have this parameter enabled unless you've taken steps to ensure that the module has the necessary resources it needs to translate names, and that it is not possible for a user to change those resources.

munge symlinks

This parameter tells rsync to modify all incoming symlinks in a way that makes them unusable but recoverable (see below). This should help protect your files from user trickery when your daemon module is writable. The default is disabled when "use chroot" is on and the inside-chroot path is "/", otherwise it is enabled.

If you disable this parameter on a daemon that is not read-only, there are tricks that a user can play with uploaded symlinks to access daemon-excluded items (if your module has any), and, if "use chroot" is off, rsync can even be tricked into showing or changing data that is outside the module's path (as access-permissions allow).

The way rsync disables the use of symlinks is to prefix each one with the string "/rsyncd-munged/". This prevents the links from being used as long as that directory does not exist. When this parameter is enabled, rsync will refuse to run if that path is a directory or a symlink to a directory. When using the "munge symlinks" parameter in a chroot area that has an inside-chroot path of "/", you should add "/rsyncd-munged/" to the exclude setting for the module so that a user can't try to create it.

Note: rsync makes no attempt to verify that any pre-existing symlinks in the module's hierarchy are as safe as you want them to be (unless, of course, it just copied in the whole hierarchy). If you setup an rsync daemon on a new area or locally add symlinks, you can manually protect your symlinks from being abused by prefixing "/rsyncd-munged/" to the start of every symlink's value. There is a perl script in the support directory of the source code named "munge-symlinks" that can be used to add or remove this prefix from your symlinks.

When this parameter is disabled on a writable module and "use chroot" is off (or the inside-chroot path is not "/"), incoming symlinks will be modified to drop a leading slash and to remove ".." path elements that rsync believes will allow a symlink to escape the module's hierarchy. There are tricky ways to work around this, though, so you had better trust your users if you choose this combination of parameters.

charset

This specifies the name of the character set in which the module's filenames are stored. If the client uses an **−−iconv** option, the daemon will use the value of the "charset" parameter regardless of the character set the client actually passed. This allows the daemon to support charset conversion in a chroot module without extra files in the chroot area, and also ensures that name-translation is done in a consistent manner. If the "charset" parameter is not set, the **−−iconv** option is refused, just as if "iconv" had been specified via "refuse options".

If you wish to force users to always use **−−iconv** for a particular module, add "no-iconv" to the "refuse options" parameter. Keep in mind that this will restrict access to your module to very new rsync clients.

max connections

This parameter allows you to specify the maximum number of simultaneous connections you will allow. Any clients connecting when the maximum has been reached will receive a message telling them to try later. The default is 0, which means no limit. A negative value disables the module. See also the "lock file" parameter.

log file

When the "log file" parameter is set to a non-empty string, the rsync daemon will log messages to the indicated file rather than using syslog. This is particularly useful on systems (such as AIX) where `syslog()` doesn't work for chrooted programs. The file is opened before `chroot()` is called, allowing it to be placed outside the transfer. If this value is set on a per-module basis instead of globally, the global log will still contain any authorization failures or config-file error messages.

If the daemon fails to open the specified file, it will fall back to using syslog and output an error about the failure. (Note that the failure to open the specified log file used to be a fatal error.)

syslog facility

This parameter allows you to specify the syslog facility name to use when logging messages from the rsync daemon. You may use any standard syslog facility name which is defined on your system. Common names are auth, authpriv, cron, daemon, ftp, kern, lpr, mail, news, security, syslog, user, uucp, local0, local1, local2, local3, local4, local5, local6 and local7. The default is daemon. This setting has no effect if the "log file" setting is a non-empty string (either set in the per-modules settings, or inherited from the global settings).

max verbosity

This parameter allows you to control the maximum amount of verbose information that you'll allow the daemon to generate (since the information goes into the log file). The default is 1, which allows the client to request one level of verbosity.

lock file

This parameter specifies the file to use to support the "max connections" parameter. The rsync daemon uses record locking on this file to ensure that the max connections limit is not exceeded for the modules sharing the lock file. The default is `/var/run/rsyncd.lock`.

read only

This parameter determines whether clients will be able to upload files or not. If "read only" is true then any attempted uploads will fail. If "read only" is false then uploads will be possible if file permissions on the daemon side allow them. The default is for all modules to be read only.

write only

This parameter determines whether clients will be able to download files or not. If "write only" is true then any attempted downloads will fail. If "write only" is false then downloads will be possible if file permissions on the daemon side allow them. The default is for this parameter to be disabled.

list This parameter determines if this module should be listed when the client asks for a listing of available modules. By setting this to false you can create hidden modules. The default is for modules to be listable.

uid This parameter specifies the user name or user ID that file transfers to and from that module should take place as when the daemon was run as root. In combination with the "gid" parameter this determines what file permissions are available. The default is uid −2, which is normally the user "nobody".

gid This parameter specifies the group name or group ID that file transfers to and from that module should take place as when the daemon was run as root. This complements the "uid" parameter. The default is gid −2, which is normally the group "nobody".

fake super
Setting "fake super = yes" for a module causes the daemon side to behave as if the **−−fake−user** command-line option had been specified. This allows the full attributes of a file to be stored without having to have the daemon actually running as root.

filter The daemon has its own filter chain that determines what files it will let the client access. This chain is not sent to the client and is independent of any filters the client may have specified. Files excluded by the daemon filter chain (**daemon-excluded** files) are treated as non-existent if the client tries to pull them, are skipped with an error message if the client tries to push them (triggering exit code 23), and are never deleted from the module. You can use daemon filters to prevent clients from downloading or tampering with private administrative files, such as files you may add to support uid/gid name translations.

The daemon filter chain is built from the "filter", "include from", "include", "exclude from", and "exclude" parameters, in that order of priority. Anchored patterns are anchored at the root of the module. To prevent access to an entire subtree, for example, "/secret", you *must* exclude everything in the subtree; the easiest way to do this is with a triple-star pattern like "/secret/***".

The "filter" parameter takes a space-separated list of daemon filter rules, though it is smart enough to know not to split a token at an internal space in a rule (e.g. "− /foo — /bar" is parsed as two rules). You may specify one or more merge-file rules using the normal syntax. Only one "filter" parameter can apply to a given module in the config file, so put all the rules you want in a single parameter. Note that per-directory merge-file rules do not provide as much protection as global rules, but they can be used to make **−−delete** work better during a client download operation if the per-dir merge files are included in the transfer and the client requests that they be used.

exclude
This parameter takes a space-separated list of daemon exclude patterns. As with the client **−−exclude** option, patterns can be qualified with "− " or "+ " to explicitly indicate exclude/include. Only one "exclude" parameter can apply to a given module. See the "filter" parameter for a description of how excluded files affect the daemon.

include
Use an "include" to override the effects of the "exclude" parameter. Only one "include" parameter can apply to a given module. See the "filter" parameter for a description of how excluded files affect the daemon.

exclude from
This parameter specifies the name of a file on the daemon that contains daemon exclude patterns, one per line. Only one "exclude from" parameter can apply to a given module; if you have multiple exclude-from files, you can specify them as a merge file in the "filter" parameter. See the

"filter" parameter for a description of how excluded files affect the daemon.

include from

Analogue of "exclude from" for a file of daemon include patterns. Only one "include from" parameter can apply to a given module. See the "filter" parameter for a description of how excluded files affect the daemon.

incoming chmod

This parameter allows you to specify a set of comma-separated chmod strings that will affect the permissions of all incoming files (files that are being received by the daemon). These changes happen after all other permission calculations, and this will even override destination-default and/or existing permissions when the client does not specify **−−perms**. See the description of the **−−chmod** rsync option and the **chmod**(1) manpage for information on the format of this string.

outgoing chmod

This parameter allows you to specify a set of comma-separated chmod strings that will affect the permissions of all outgoing files (files that are being sent out from the daemon). These changes happen first, making the sent permissions appear to be different than those stored in the filesystem itself. For instance, you could disable group write permissions on the server while having it appear to be on to the clients. See the description of the **−−chmod** rsync option and the **chmod**(1) manpage for information on the format of this string.

auth users

This parameter specifies a comma and space-separated list of usernames that will be allowed to connect to this module. The usernames do not need to exist on the local system. The usernames may also contain shell wildcard characters. If "auth users" is set then the client will be challenged to supply a username and password to connect to the module. A challenge response authentication protocol is used for this exchange. The plain text usernames and passwords are stored in the file specified by the "secrets file" parameter. The default is for all users to be able to connect without a password (this is called "anonymous rsync").

See also the "CONNECTING TO AN RSYNC DAEMON OVER A REMOTE SHELL PROGRAM" section in **rsync**(1) for information on how handle an rsyncd.conf−level username that differs from the remote-shell-level username when using a remote shell to connect to an rsync daemon.

secrets file

This parameter specifies the name of a file that contains the username:password pairs used for authenticating this module. This file is only consulted if the "auth users" parameter is specified. The file is line based and contains username:password pairs separated by a single colon. Any line starting with a hash (#) is considered a comment and is skipped. The passwords can contain any characters but be warned that many operating systems limit the length of passwords that can be typed at the client end, so you may find that passwords longer than 8 characters don't work.

There is no default for the "secrets file" parameter, you must choose a name (such as `/etc/rsyncd.secrets`). The file must normally not be readable by "other"; see "strict modes".

strict modes

This parameter determines whether or not the permissions on the secrets file will be checked. If "strict modes" is true, then the secrets file must not be readable by any user ID other than the one that the rsync daemon is running under. If "strict modes" is false, the check is not performed. The default is true. This parameter was added to accommodate rsync running on the Windows operating system.

hosts allow

This parameter allows you to specify a list of patterns that are matched against a connecting clients hostname and IP address. If none of the patterns match then the connection is rejected.

Each pattern can be in one of five forms:

o a dotted decimal IPv4 address of the form a.b.c.d, or an IPv6 address of the form a:b:c::d:e:f. In this case the incoming machine's IP address must match exactly.

o an address/mask in the form ipaddr/n where ipaddr is the IP address and n is the number of one bits in the netmask. All IP addresses which match the masked IP address will be allowed in.

o an address/mask in the form ipaddr/maskaddr where ipaddr is the IP address and maskaddr is the netmask in dotted decimal notation for IPv4, or similar for IPv6, e.g. ffff:ffff:ffff:ffff:: instead of /64. All IP addresses which match the masked IP address will be allowed in.

o a hostname. The hostname as determined by a reverse lookup will be matched (case insensitive) against the pattern. Only an exact match is allowed in.

o a hostname pattern using wildcards. These are matched using the same rules as normal unix filename matching. If the pattern matches then the client is allowed in.

Note IPv6 link-local addresses can have a scope in the address specification:

```
fe80::1%link1
fe80::%link1/64
fe80::%link1/ffff:ffff:ffff:ffff::
```

You can also combine "hosts allow" with a separate "hosts deny" parameter. If both parameters are specified then the "hosts allow" parameter is checked first and a match results in the client being able to connect. The "hosts deny" parameter is then checked and a match means that the host is rejected. If the host does not match either the "hosts allow" or the "hosts deny" patterns then it is allowed to connect.

The default is no "hosts allow" parameter, which means all hosts can connect.

hosts deny

This parameter allows you to specify a list of patterns that are matched against a connecting clients hostname and IP address. If the pattern matches then the connection is rejected. See the "hosts allow" parameter for more information.

The default is no "hosts deny" parameter, which means all hosts can connect.

ignore errors

This parameter tells rsyncd to ignore I/O errors on the daemon when deciding whether to run the delete phase of the transfer. Normally rsync skips the **−−delete** step if any I/O errors have occurred in order to prevent disastrous deletion due to a temporary resource shortage or other I/O error. In some cases this test is counter productive so you can use this parameter to turn off this behavior.

ignore nonreadable

This tells the rsync daemon to completely ignore files that are not readable by the user. This is useful for public archives that may have some non-readable files among the directories, and the sysadmin doesn't want those files to be seen at all.

transfer logging

This parameter enables per-file logging of downloads and uploads in a format somewhat similar to that used by ftp daemons. The daemon always logs the transfer at the end, so if a transfer is aborted, no mention will be made in the log file.

If you want to customize the log lines, see the “log format” parameter.

log format

This parameter allows you to specify the format used for logging file transfers when transfer logging is enabled. The format is a text string containing embedded single-character escape sequences prefixed with a percent (%) character. An optional numeric field width may also be specified between the percent and the escape letter (e.g. “**%−50n %8l %07p**”).

The default log format is “%o %h [%a] %m (%u) %f %l”, and a “%t [%p] ” is always prefixed when using the “log file” parameter. (A perl script that will summarize this default log format is included in the rsync source code distribution in the “support” subdirectory: rsyncstats.)

The single-character escapes that are understood are as follows:

- %a the remote IP address
- %b the number of bytes actually transferred
- %B the permission bits of the file (e.g. rwxrwxrwt)
- %c the total size of the block checksums received for the basis file (only when sending)
- %f the filename (long form on sender; no trailing “/”)
- %G the gid of the file (decimal) or “DEFAULT”
- %h the remote host name
- %i an itemized list of what is being updated
- %l the length of the file in bytes
- %L the string “ −> SYMLINK”, “ => HARDLINK”, or “” (where **SYMLINK** or **HARDLINK** is a filename)
- %m the module name
- %M the last-modified time of the file
- %n the filename (short form; trailing “/” on dir)
- %o the operation, which is “send”, “recv”, or “del.” (the latter includes the trailing period)
- %p the process ID of this rsync session
- %P the module path
- %t the current date time
- %u the authenticated username or an empty string
- %U the uid of the file (decimal)

For a list of what the characters mean that are output by “%i”, see the **−−itemize−changes** option in the rsync manpage.

Note that some of the logged output changes when talking with older rsync versions. For instance, deleted files were only output as verbose messages prior to rsync 2.6.4.

timeout

This parameter allows you to override the clients choice for I/O timeout for this module. Using this parameter you can ensure that rsync won’t wait on a dead client forever. The timeout is specified in seconds. A value of zero means no timeout and is the default. A good choice for anonymous rsync daemons may be 600 (giving a 10 minute timeout).

refuse options

This parameter allows you to specify a space-separated list of rsync command line options that will be refused by your rsync daemon. You may specify the full option name, its one-letter abbreviation, or a wild-card string that matches multiple options. For example, this would refuse **−−checksum** (**−c**) and all the various delete options:

```
    refuse options = c delete
```

The reason the above refuses all delete options is that the options imply **−−delete**, and implied options are refused just like explicit options. As an additional safety feature, the refusal of “delete” also refuses **remove-source-files** when the daemon is the sender; if you want the latter without the former, instead refuse “delete−*” — that refuses all the delete modes without affecting **−−remove−source−files**.

When an option is refused, the daemon prints an error message and exits. To prevent all compression when serving files, you can use “dont compress = *” (see below) instead of “refuse options = compress” to avoid returning an error to a client that requests compression.

dont compress

This parameter allows you to select filenames based on wildcard patterns that should not be compressed when pulling files from the daemon (no analogous parameter exists to govern the pushing of files to a daemon). Compression is expensive in terms of CPU usage, so it is usually good to not try to compress files that won’t compress well, such as already compressed files.

The “dont compress” parameter takes a space-separated list of case-insensitive wildcard patterns. Any source filename matching one of the patterns will not be compressed during transfer.

See the **−−skip−compress** parameter in the **rsync**(1) manpage for the list of file suffixes that are not compressed by default. Specifying a value for the “dont compress” parameter changes the default when the daemon is the sender.

pre-xfer exec, **post-xfer exec**

You may specify a command to be run before and/or after the transfer. If the **pre-xfer exec** command fails, the transfer is aborted before it begins.

The following environment variables will be set, though some are specific to the pre-xfer or the post-xfer environment:

- **RSYNC_MODULE_NAME**: The name of the module being accessed.
- **RSYNC_MODULE_PATH**: The path configured for the module.
- **RSYNC_HOST_ADDR**: The accessing host’s IP address.
- **RSYNC_HOST_NAME**: The accessing host’s name.
- **RSYNC_USER_NAME**: The accessing user’s name (empty if no user).
- **RSYNC_PID**: A unique number for this transfer.
- **RSYNC_REQUEST**: (pre-xfer only) The module/path info specified by the user (note that the user can specify multiple source files, so the request can be something like “mod/path1 mod/path2”, etc.).
- **RSYNC_ARG#**: (pre-xfer only) The pre-request arguments are set in these numbered values. RSYNC_ARG0 is always “rsyncd”, and the last value contains a single period.
- **RSYNC_EXIT_STATUS**: (post-xfer only) the server side’s exit value. This will be 0 for a successful run, a positive value for an error that the server generated, or a −1 if rsync failed to exit properly. Note that an error that occurs on the client side does not currently get sent to the server side, so this is not the final exit status for the whole transfer.

- **RSYNC_RAW_STATUS**: (post-xfer only) the raw exit value from `waitpid()` .

Even though the commands can be associated with a particular module, they are run using the permissions of the user that started the daemon (not the module's uid/gid setting) without any chroot restrictions.

AUTHENTICATION STRENGTH

The authentication protocol used in rsync is a 128 bit MD4 based challenge response system. This is fairly weak protection, though (with at least one brute-force hash-finding algorithm publicly available), so if you want really top-quality security, then I recommend that you run rsync over ssh. (Yes, a future version of rsync will switch over to a stronger hashing method.)

Also note that the rsync daemon protocol does not currently provide any encryption of the data that is transferred over the connection. Only authentication is provided. Use ssh as the transport if you want encryption.

Future versions of rsync may support SSL for better authentication and encryption, but that is still being investigated.

EXAMPLES

A simple rsyncd.conf file that allow anonymous rsync to a ftp area at `/home/ftp` would be:

```
[ftp]
        path = /home/ftp
        comment = ftp export area
```

A more sophisticated example would be:

```
uid = nobody
gid = nobody
use chroot = yes
max connections = 4
syslog facility = local5
pid file = /var/run/rsyncd.pid

[ftp]
        path = /var/ftp/./pub
        comment = whole ftp area (approx 6.1 GB)

[sambaftp]
        path = /var/ftp/./pub/samba
        comment = Samba ftp area (approx 300 MB)

[rsyncftp]
        path = /var/ftp/./pub/rsync
        comment = rsync ftp area (approx 6 MB)

[sambawww]
        path = /public_html/samba
        comment = Samba WWW pages (approx 240 MB)

[cvs]
        path = /data/cvs
        comment = CVS repository (requires authentication)
        auth users = tridge, susan
        secrets file = /etc/rsyncd.secrets
```

The /etc/rsyncd.secrets file would look something like this:

```
    tridge:mypass
    susan:herpass
```

FILES

/etc/rsyncd.conf or rsyncd.conf

SEE ALSO

rsync(1)

DIAGNOSTICS

BUGS

Please report bugs! The rsync bug tracking system is online at http://rsync.samba.org/

VERSION

This man page is current for version 3.0.6 of rsync.

CREDITS

rsync is distributed under the GNU public license. See the file COPYING for details.

The primary ftp site for rsync is ftp://rsync.samba.org/pub/rsync.

A WEB site is available at http://rsync.samba.org/

We would be delighted to hear from you if you like this program.

This program uses the zlib compression library written by Jean-loup Gailly and Mark Adler.

THANKS

Thanks to Warren Stanley for his original idea and patch for the rsync daemon. Thanks to Karsten Thygesen for his many suggestions and documentation!

AUTHOR

rsync was written by Andrew Tridgell and Paul Mackerras. Many people have later contributed to it.

Mailing lists for support and development are available at http://lists.samba.org