rsync (download)

Remote file copy - Synchronize file trees across local disks, directories or across a network.

Syntax

```
 Local file to Local file:
      rsync [option]... Source [Source]... Dest

 Local to Remote:
      rsync [option]... Source [Source]... [user@]host:Dest   #

      rsync [option...] [user@]host::Source... [Dest]

      rsync [option...] rsync://[user@]host[:PORT]/Source... [Dest]

 Remote to Local:
      rsync [option]... [user@]host:Source... [Dest]          #

      rsync [option]... [user@]host::Dest

      rsync [option]... rsync://[user@]host[:PORT]/Dest

 # = via remote shell rather than the rsync daemon
```

rsync is a program that behaves in much the same way that rcp does, but has many more options and uses the rsync remote-update protocol to greatly speed up file transfers when the destination file already exists. Rsync is widely used for backups and mirroring and as an improved copy command for everyday use.

Rsync finds files that need to be transferred using a "quick check" algorithm (by default) that looks for files that have changed in size or in last-modified time. Any changes in the other preserved attributes (as requested by options) are made on the destination file directly when the quick check indicates that the file's data does not need to be updated.

Some of the additional features of rsync are:

- Support for copying links, devices, owners, groups and permissions
- Exclude and exclude-from options similar to GNU tar
- A CVS exclude mode for ignoring the same files that CVS would ignore
- Can use any transparent remote shell, including rsh or ssh
- Does not require root privileges
- Pipelining of file transfers to minimize latency costs
- Support for anonymous or authenticated rsync servers (ideal for mirroring)

Usage

You use rsync in the same way you use rcp. You must specify a source and a destination, one of which can be remote.

Perhaps the best way to explain the syntax is some examples:

```
rsync -t *.c foo:src/
```

this would transfer all files matching the pattern *.c from the current directory to the directory src on the machine foo. If any of the files already exist on the remote system then the rsync remote-update protocol is used to update the file by sending only the differences. See the tech report for details.

```
rsync -avz foo:src/bar /data/tmp
```

This would recursively transfer all files from the directory src/bar on the machine foo into the /data/tmp/bar directory on the local machine.
The files are transferred in "archive" mode, which ensures that symbolic links, devices, attributes, permissions, ownerships etc are preserved in the transfer.
Additionally, compression will be used to reduce the size of data portions of the transfer.

```
rsync -avz foo:src/bar/ /data/tmp
```

a trailing slash on the source changes this behavior to transfer all files from the directory src/bar on the machine foo into the /data/tmp/.
A trailing / on a source name means "copy the contents of this directory". Without a trailing slash it means "copy the directory".
This difference becomes particularly important when using the --delete option.

You can also use rsync in local-only mode, where both the source and destination don't have a ':' in the name. In this case it behaves like an improved copy command.

```
rsync somehost.mydomain.com::
```

this would list all the anonymous rsync modules available on the host somehost.mydomain.com. (See the following section for more details.)

Connecting to a RSYNC Server

It is also possible to use rsync without using rsh or ssh as the transport. In this case you will connect to a remote rsync server running on TCP port 873.

You can establish the connection via a web proxy by setting the environment variable RSYNC_PROXY to a hostname:port pair pointing to your web proxy.
Note that your web proxy's configuration must allow proxying to port 873.

Using rsync in this way is the same as using it with rsh or ssh except that:

- You use a double colon :: instead of a single colon to separate the hostname from the path.
- The first word of the "path" is actually a module name.
- The remote daemon might print a message of the day when you connect.
- If you specify no path name on the remote daemon then the list of accessible paths on the daemon will be shown.
- If you specify no local destination then a listing of the specified files on the remote server is provided.
- Do not specify the `--rsh (-e)` option.

Some paths on the remote server might require authentication. If so then you will receive a password prompt when you connect.
You can avoid the password prompt by setting the environment variable RSYNC_PASSWORD to the password you want to use or using the `--password-file` option.
This can be useful when scripting rsync.

WARNING: On some systems environment variables are visible to all users. On those systems using `--password-file` is recommended.

Running an RSYNC Server

An rsync server is configured using a config file which by default is called `/etc/rsyncd.conf`. Please see the rsyncd.conf(5) man page for more information.

EXAMPLES

To Backup the home directory using a cron job:

```
rsync -Cavz . ss64:backup
```

Run the above over a PPP link to a duplicate directory on machine "ss64".

To synchronize samba source trees use the following Makefile targets:

get:
```
rsync -avuzb --exclude '*~' samba:samba/ .
```

put:
```
rsync -Cavuzb . samba:samba/
```

sync: `get put`

this allows me to sync with a CVS directory at the other end of the link. I then do cvs operations on the remote machine, which saves a lot of time as the remote cvs protocol isn't very efficient.

I mirror a directory between my "old" and "new" ftp sites with the command

```
rsync -az -e ssh --delete ~ftp/pub/samba/ nimbus:"~ftp/pub/tridge/samba"
```

this is launched from cron every few hours.

OPTIONS SUMMARY

Here is a short summary of the options available in rsync.
Please refer to the FULL List of OPTIONS for a complete description.

What to copy:
```
 -r, --recursive             recurse into directories
```

```
 -R, --relative              use relative path names
     --exclude=PATTERN       Exclude files matching PATTERN
     --exclude-from=FILE     Read exclude patterns from FILE
 -I, --ignore-times          Don't exclude files that match length and time
     --size-only             only use file size when determining if a file should be transferred
 -@  --modify-window=NUM     Timestamp window (seconds) for file match (default=0)
     --include=PATTERN       Don't exclude files matching PATTERN
     --include-from=FILE     Read include patterns from FILE

How to copy it:
 -n, --dry-run               Perform a trial run with no changes made
 -l, --links                 Copy symlinks as symlinks
 -L, --copy-links            Transform symlink into referent file/dir
     --copy-unsafe-links     Only "unsafe" symlinks are transformed
     --safe-links            Ignore links outside the destination tree
     --munge-links           Munge symlinks to make them safer
 -H, --hard-links            Preserve hard links
     --devices               preserve device files (super-user only)
     --specials              preserve special files
 -D, --devices --specials    Preserve devices (super-user only) +files
 -g, --group                 Preserve group
 -o, --owner                 Preserve owner (super-user only)
 -p, --perms                 Preserve permissions
     --remove-source-files   Sender removes synchronized files (non-dir)
 -t, --times                 Preserve times
 -S, --sparse                Handle sparse files efficiently
 -x, --one-file-system       Don't cross filesystem boundaries
 -B, --block-size=SIZE       Force a fixed checksum block-size (default 700)
 -e, --rsh=COMMAND           Specify rsh replacement
     --rsync-path=PATH       Specify path to rsync on the remote machine
     --numeric-ids           Don't map uid/gid values by user/group name
     --timeout=SECONDS       Set IO timeout in seconds
 -W, --whole-file            Copy whole files, no incremental checks

Destination options:
 -a, --archive               Archive mode equals -rlptgoD (no -H,-A,-X)
 -b, --backup                Make backups (see --suffix & --backup-dir)
```

```
     --backup-dir=DIR        Make backups into this directory
 -z, --compress              Compress file data during the transfer
 -c, --checksum              Skip based on checksum, not mod-time & size
 -C, --cvs-exclude           Auto ignore files in the same way CVS does
     --existing              Only update files that already exist
     --delete                Delete files that don't exist on the sending side
     --delete-excluded       also delete excluded files on the receiving side
     --delete-after          Receiver deletes after transfer, not during
     --force                 Force deletion of directories even if not empty
     --ignore-errors         Delete even if there are IO errors
     --max-delete=NUM        Don't delete more than NUM files
     --log-file-format=FMT   Log file transfers using specified format
     --partial               Keep partially transferred files
     --progress              Show progress during transfer
 -P                          equivalent to --partial --progress
     --stats                 Give some file transfer stats
 -T  --temp-dir=DIR          Create temporary files in directory DIR
     --compare-dest=DIR      also compare destination files relative to DIR
 -u, --update                update only (don't overwrite newer files)

Misc Others:
     --address=ADDRESS       bind to the specified address
     --blocking-io           Use blocking IO for the remote shell
     --bwlimit=KBPS          Limit I/O bandwidth, KBytes per second
     --config=FILE           Specify alternate rsyncd.conf file (daemon)
     --daemon                Run as a rsync daemon
     --no-detach             Do not detach from the parent (daemon)
     --password-file=FILE    Get daemon-access password from FILE
     --port=PORT             Specify alternate rsyncd port number
 -f, --read-batch=FILE       Read batch file
 -F, --write-batch=FILE      Write batch file
     --version               Print version number
 -v, --verbose               Increase verbosity
 -q, --quiet                 Decrease verbosity
 -4, --ipv4                  Prefer IPv4
 -6, --ipv6                  Prefer IPv6
 -h, --help                  show this help screen
```

Tips on how to use each of the options above can be found in the
FULL List of OPTIONS and Exit Values

EXCLUDE PATTERNS

The exclude and include patterns specified to rsync allow for flexible selection of which files to transfer and which files to skip.

rsync builds an ordered list of include/exclude options as specified on the command line. When a filename is encountered, rsync checks the name against each exclude/include pattern in turn. The first matching pattern is acted on.
If it is an exclude pattern, then that file is skipped.
If it is an include pattern then that filename is not skipped.
If no matching include/exclude pattern is found then the filename is not skipped.

Note that when used with -r (which is implied by -a), every subcomponent of every path is visited from top down, so include/exclude patterns get applied recursively to each subcomponent.

Note also that the --include and --exclude options take one pattern each.
To add multiple patterns use the `--include-from` and `--exclude-from` options or multiple `--include` and `--exclude` options.

The patterns can take several forms. The rules are:

if the pattern starts with a `/` then it is matched against the start of the filename, otherwise it is matched against the end of the filename.
Thus "/foo" would match a file called "foo" at the base of the tree. On the other hand, "foo" would match any file called "foo" anywhere in the tree
because the algorithm is applied recursively from top down; it behaves as if each path component gets a turn at being the end of the file name.

if the pattern ends with a `/` then it will only match a directory, not a file, link or device.

if the pattern contains a wildcard character from the set `*?[` then expression matching is applied using the shell filename matching rules.
Otherwise a simple string match is used.

if the pattern includes a double asterisk "`**`" then all wildcards in the pattern will match slashes, otherwise they will stop at slashes.

if the pattern contains a `/` (not counting a trailing /) then it is matched against the full filename, including any leading directory.

If the pattern doesn't contain a `/` then it is matched only against the final component of the filename. Again, remember that the algorithm is applied recursively so "full filename" can actually be any portion of a path.

if the pattern starts with "`+ `" (a plus followed by a space) then it is always considered an include pattern, even if specified as part of an exclude option. The "+ " part is discarded before matching.

if the pattern starts with "`- `" (a minus followed by a space) then it is always considered an exclude pattern, even if specified as part of an include option. The "- " part is discarded before matching.

if the pattern is a single exclamation mark `!` then the current include/exclude list is reset, removing all previously defined patterns.

The +/- rules are most useful in exclude lists, allowing you to have a single exclude list that contains both include and exclude options.

If you end an exclude list with `--exclude '*'`, note that since the algorithm is applied recursively that unless you explicitly include parent directories of files you want to include then the algorithm will stop at the parent directories and never see the files below them. To include all directories, use `--include '*/'` before the `--exclude '*'`.

Some exclude/include examples:

```
# --exclude "*.o"   would exclude all filenames matching *.o
# --exclude "/foo"  would exclude a file in the base directory called foo
# --exclude "foo/"  would exclude any directory called foo.
# --exclude "/foo/*/bar"  would exclude any file called bar two levels below a
                          base directory called foo.
# --exclude "/foo/**/bar" would exclude any file called bar two or more levels below
                          a base directory called foo.
# --include "*/" --include "*.c" --exclude "*"
                          would include all directories
                          and C source files
# --include "foo/" --include "foo/bar.c" --exclude "*"
                   would include only foo/bar.c (the foo/ directory must be
                   explicitly included or it would be excluded by the "*")
```

Batch Mode

The following call generates 4 files that encapsulate the information for synchronizing the contents of target_dir with the updates found in src_dir

```
$ rsync -F [other rsync options here] \
/somewhere/src_dir /somewhere/target_dir

The generated files are labeled with a common timestamp:

# rsync_argvs. command-line arguments
# rsync_flist. rsync internal file metadata
# rsync_csums. rsync checksums
# rsync_delta. data blocks for file update & change
```

Symbolic Links

Three basic behaviours are possible when rsync encounters a symbolic link in
the source directory.

By default, symbolic links are not transferred at all.
A message "skipping non-regular" file is emitted for any symlinks that exist.

If --links is specified, then symlinks are recreated with the same target
on the destination. Note that --archive implies --links.

If --copy-links is specified, then symlinks are "collapsed" by copying their referent,
rather than the symlink.

rsync also distinguishes "safe" and "unsafe" symbolic links.
An example where this might be used is a web site mirror that wishes ensure the
rsync module they copy does not include symbolic links to /etc/passwd in the public
section of the site. Using --copy-unsafe-links will cause any links to be copied
as the file they point to on the destination.
Using --safe-links will cause unsafe links to be ommitted altogether.

Diagnostics

rsync occasionally produces error messages that can seem a little cryptic.
The one that seems to cause the most confusion is "protocol version mismatch - is your shell clean?".

This message is usually caused by your startup scripts or remote shell facility producing unwanted garbage on the stream that rsync is using for its transport. The way to diagnose this problem is to run your remote shell like this:

```
rsh remotehost /bin/true > out.dat
```

then look at out.dat. If everything is working correctly then out.dat should be a zero length file. If you are getting the above error from rsync then you will probably find that out.dat contains some text or data.
Look at the contents and try to work out what is producing it.
The most common cause is incorrectly configured shell startup scripts (such as .cshrc or .profile) that contain output statements for non-interactive logins.

If you are having trouble debugging include and exclude patterns, then try specifying the -vv option.
At this level of verbosity rsync will show why each individual file is included or
excluded.

Setup

See the file README for installation instructions.

Once installed you can use rsync to any machine that you can use rsh to.
rsync uses rsh for its communications, unless both the source and destination are local.

You can also specify an alternative to rsh, either by using the -e command line
option, or by setting the RSYNC_RSH environment variable.

One common substitute is to use ssh, which offers a high degree of security.

Note that rsync must be installed on both the source and destination machines.

Environment Variables

```
CVSIGNORE
    The CVSIGNORE environment variable supplements any ignore patterns in .cvsignore files.
    See the --cvs-exclude option for more details.

RSYNC_RSH
    The RSYNC_RSH environment variable allows you to override the default shell used as
```

```
        the transport for rsync. This can be used instead of the -e option.

    RSYNC_PROXY
        The RSYNC_PROXY environment variable allows you to redirect your rsync client to
        use a web proxy when connecting to a rsync daemon.
        You should set RSYNC_PROXY to a hostname:port pair.

    RSYNC_PASSWORD
        Setting RSYNC_PASSWORD to the required password allows you to run authenticated
        rsync connections to a rsync daemon without user intervention.
        Note that this does not supply a password to a shell transport such as ssh.

    USER or LOGNAME
        The USER or LOGNAME environment variables are used to determine the default
        username sent to a rsync server.

    HOME
        The HOME environment variable is used to find the user's default .cvsignore file.
```

Files

```
    /etc/rsyncd.conf
```

“And yet I do observe that audiences which used to be deeply affected by the inspiring sternness of the music of Livius and Naevius, now leap up and twist their necks and turn their eyes in time with our modern tunes” ~ Cicero (De Legibus II.39 c. 50 BCE) on the evils of modern music.

Related linux commands:

Grsync - GUI for rsync (how to install)
rsyncd.conf(5)
rsnapshot - Save multiple backups with rsync
rcp - Copy files between two machines.
cp - Copy one or more files to another location
install - Copy files and set attributes
dd - Data Dump - convert and copy a file (use for RAW storage)

remsync - Synchronize remote files via email
Equivalent Windows command: ROBOCOPY - Robust File and Folder Copy

---