rsync

rsync (https://rsync.samba.org/) is an open source utility that provides fast incremental file transfer.

Contents

Related articles

System backup

Synchronization and backup programs

Installation

Install the **rsync (https://www.archlinux.org/packages/?name=rsync)** package.

rsync must be installed on both the source and the destination machine.

Front-ends

- **Grsync** — GTK+ front-end.

  http://www.opbyte.it/grsync/ || **grsync (https://www.archlinux.org/packages/?name=grsync)**

- **gutback** — rsync wrapper written in Shell.

  https://github.com/gutenye/gutbackup || **gutbackup (https://aur.archlinux.org/packages/gutbackup/)**[AUR]

- **JotaSync** — Java Swing GUI for rsync with integrated scheduler.

https://trixon.se/projects/jotasync/ || jotasync (https://aur.archlinux.org/packages/jotasync/)[AUR]

- **luckyBackup** — Qt front-end written in C++.

  http://luckybackup.sourceforge.net/index.html || luckybackup (https://aur.archlinux.org/packages/luckybackup/)[AUR]

As a cp alternative

rsync can be used as an advanced alternative for the `cp` command, especially for copying larger files:

```
$ rsync -P source destination
```

The `-P` option is the same as `--partial --progress`, which keeps partially transferred files and shows a progress bar during transfer.

You may want to use the `-r` / `--recursive` option to recurse into directories.

Files can be copied locally as with cp, but the motivating purpose of rsync is to copy files remotely, i.e. between two different hosts. Remote locations can be specified with a host-colon syntax:

```
$ rsync source host:destination
```

or

```
$ rsync host:source destination
```

Network file transfers use the **SSH** protocol by default and `host` can be a real hostname or a predefined profile/alias from `.ssh/config`.

Whether transferring files locally or remotely, rsync first creates an index of block checksums of each source file. This index is used to find any identical blocks of data which might exist in the destination. Such blocks are used in-place, rather than being copied from the source. This can greatly accelerate the synchronization of large files with small changes. For more information, see **official documentation (https://rsync.samba.org/documentation.html)**, **how rsync works (https://rsync.samba.org/how-rsync-works.html)**.

Trailing slash caveat

Arch by default uses GNU cp (part of **GNU coreutils (https://www.archlinux.org/packages/?name=coreutils)**). However, rsync follows the convention of BSD cp, which gives special treatment to source directories with a trailing slash "/". Although

```
$ rsync -r source destination
```

creates a directory "destination/source" with the contents of "source", the command

```
$ rsync -r source/ destination
```

copies all of the files in "source/" directly into "destination", with no intervening subdirectory - just as if you had invoked it as

```
$ rsync -r source/. destination
```

This behavior is different from that of GNU cp, which treats "source" and "source/" identically (but not "source/."). Also, some shells automatically append the trailing slash when tab-completing directory names. Because of these factors, there can be a tendency among new or occasional rsync users to forget about rsync's different behavior, and inadvertently create a mess or even overwrite important files by leaving the trailing slash on the command line.

Thus it can be prudent to use a wrapper script to automatically remove trailing slashes before invoking rsync:

```
#!/bin/zsh
new_args=();
for i in "$@"; do
    case $i in /) i=/;; */) i=${i%/};; esac
    new_args+=$i;
done
exec rsync "${(@)new_args}"
```

This script can be put somewhere in the path, and aliased to rsync in the shell init file.

As a backup utility

The rsync protocol can easily be used for backups, only transferring files that have changed since the last backup. This section describes a very simple scheduled backup script using rsync, typically used for copying to removable media.

Automated backup

For the sake of this example, the script is created in the `/etc/cron.daily` directory, and will be run on a daily basis if a cron **daemon** is installed and properly configured. Configuring and using **cron** is outside the scope of this article.

First, create a script containing the appropriate command options:

```
/etc/cron.daily/backup
```

```
#!/bin/bash
rsync -a --delete --quiet /folder/to/backup /location/of/backup
```

`-a`
: indicates that files should be archived, meaning that most of their characteristics are preserved (but **not** ACLs, hard links or extended attributes such as capabilities)

`--delete`

means files deleted on the source are to be deleted on the backup as well

Here, `/folder/to/backup` should be changed to what needs to be backed-up (`/home` , for example) and `/location/to/backup` is where the backup should be saved (`/media/disk` , for instance).

Finally, the script must be executable:

```
# chmod +x /etc/cron.daily/backup
```

Automated backup with SSH

If backing-up to a remote host using SSH, use this script instead:

```
/etc/cron.daily/backup
```

```
#!/bin/bash
rsync -a --delete --quiet -e ssh /folder/to/backup remoteuser@remotehost:/location/of/backup
```

`-e ssh`
: tells rsync to use SSH

`remoteuser`
: is the user on the host `remotehost`

`-a`
: groups all these options `-rlptgoD` (recursive, links, perms, times, group, owner, devices)

Automated backup with NetworkManager

This script starts a backup when network connection is established.

First, create a script containing the appropriate command options:

```
/etc/NetworkManager/dispatcher.d/backup
```

```
#!/bin/bash

if [ x"$2" = "xup" ] ; then
        rsync --force --ignore-errors -a --delete --bwlimit=2000 --files-from=files.rsync /folder/to/backup /location/to/backup
fi
```

`-a`
: group all this options `-rlptgoD` recursive, links, perms, times, group, owner, devices

`--files-from`
: read the relative path of */folder/to/backup* from this file

`--bwlimit`
: limit I/O bandwidth; KBytes per second

Also, the script must have write permission for owner (root, of course) only (see **NetworkManager#Network services with NetworkManager dispatcher** for details).

Automated backup with systemd and inotify

Note:

- Due to the limitations of inotify and systemd (see **this question and answer (http://www.quora.com/Linux-Kernel/Inotify-monitoring-of-directories-is-not-recursive-Is-there-any-specific-reason-for-this-design-in-Linux-kernel)**), recursive filesystem monitoring is not possible. Although you can watch a directory and its contents, it will not recurse into subdirectories and watch the contents of them; you must explicitly specify every directory to watch, even if that directory is a child of an already watched directory.
- This setup is based on a **systemd/User** instance.

Instead of running time interval backups with time based schedules, such as those implemented in **cron**, it is possible to run a backup every time one of the files you are backing up changes. `systemd.path` units use `inotify` to monitor the filesystem, and can be used in conjunction with `systemd.service` files to start any process (in this case your **rsync** backup) based on a filesystem event.

First, create the `systemd.path` file that will monitor the files you are backing up:

~/.config/systemd/user/backup.path

```
[Unit]
Description=Checks if paths that are currently being backed up have changed

[Path]
PathChanged=%h/documents
PathChanged=%h/music

[Install]
WantedBy=default.target
```

Then create a `systemd.service` file that will be activated when it detects a change. By default a service file of the same name as the path unit (in this case `backup.path`) will be activated, except with the `.service` extension instead of `.path` (in this case `backup.service`).

Note: If you need to run multiple rsync commands, use `Type=oneshot`. This allows you to specify multiple `ExecStart=` parameters, one for each **rsync** command, that will be executed. Alternatively, you can simply write a script to perform all of your backups, just like **cron** scripts.

~/.config/systemd/user/backup.service

```
[Unit]
Description=Backs up files

[Service]
ExecStart=/usr/bin/rsync %h/./documents %h/./music -CERrltm --delete ubuntu:
```

Now all you have to do is **start**/enable `backup.path` like a normal systemd service and it will start monitoring file changes and automatically starting `backup.service`.

Differential backup on a week

This is a useful option of rsync, resulting in a full backup (on each run) and keeping a differential backup copy of changed files only in a separate directory for each day of a week.

First, create a script containing the appropriate command options:

/etc/cron.daily/backup

```
#!/bin/bash

DAY=$(date +%A)

if [ -e /location/to/backup/incr/$DAY ] ; then
  rm -fr /location/to/backup/incr/$DAY
fi

rsync -a --delete --quiet --inplace --backup --backup-dir=/location/to/backup/incr/$DAY /folder/to/backup/ /location/to/backup/full/
```

`--inplace`
: implies `--partial` update destination files in-place

Snapshot backup

The same idea can be used to maintain a tree of snapshots of your files. In other words, a directory with date-ordered copies of the files. The copies are made using hardlinks, which means that only files that did change will occupy space. Generally speaking, this is the idea behind Apple's TimeMachine.

This basic script is easy to implement and creates quick incremental snapshots using the `--link-dest` option to hardlink unchanged files:

/usr/local/bin/snapbackup.sh

```
#!/bin/bash

# Basic snapshot-style rsync backup script

# Config
OPT="-aPh"
```

```
LINK="--link-dest=/snapshots/username/last/"
SRC="/home/username/files/"
SNAP="/snapshots/username/"
LAST="/snapshots/username/last"
date=`date "+%Y-%b-%d:_%T"`

# Run rsync to create snapshot
rsync $OPT $LINK $SRC ${SNAP}$date

# Remove symlink to previous snapshot
rm -f $LAST

# Create new symlink to latest snapshot for the next backup to hardlink
ln -s ${SNAP}$date $LAST
```

There must be a symlink to a full backup already in existence as a target for `--link-dest`. If the most recent snapshot is deleted, the symlink will need to be recreated to point to the most recent snapshot. If `--link-dest` does not find a working symlink, rsync will proceed to copy all source files instead of only the changes.

A more sophisticated version keeps an up-to-date full backup `$SNAP/latest` and in case a certain number of files has changed since the last full backup, it creates a snapshot `$SNAP/$DATETAG` of the current full-backup utilizing `cp -al` to hardlink unchanged files:

/usr/local/bin/rsnapshot.sh

```
#!/bin/bash

## my own rsync-based snapshot-style backup procedure
## (cc) marcio rps AT gmail.com

# config vars

SRC="/home/username/files/" #dont forget trailing slash!
SNAP="/snapshots/username"
OPTS="-rltgoi --delay-updates --delete --chmod=a-w"
MINCHANGES=20

# run this process with real low priority
```

```
ionice -c 3 -p $$
renice +12  -p $$

# sync

rsync $OPTS $SRC $SNAP/latest >> $SNAP/rsync.log

# check if enough has changed and if so
# make a hardlinked copy named as the date

COUNT=$( wc -l $SNAP/rsync.log|cut -d" " -f1 )
if [ $COUNT -gt $MINCHANGES ] ; then
        DATETAG=$(date +%Y-%m-%d)
        if [ ! -e $SNAP/$DATETAG ] ; then
                cp -al $SNAP/latest $SNAP/$DATETAG
                chmod u+w $SNAP/$DATETAG
                mv $SNAP/rsync.log $SNAP/$DATETAG
               chmod u-w $SNAP/$DATETAG
         fi
fi
```

To make things really, really simple this script can be run from a **systemd/Timers** unit.

Full system backup

This section is about using *rsync* to transfer a copy of the entire `/` tree, excluding a few selected folders. This approach is considered to be better than **disk cloning** with `dd` since it allows for a different size, partition table and filesystem to be used, and better than copying with `cp -a` as well, because it allows greater control over file permissions, attributes, **Access Control Lists** and **extended attributes**.

rsync will work even while the system is running, but files changed during the transfer may or may not be transferred, which can cause undefined behavior of some programs using the transferred files.

This approach works well for migrating an existing installation to a new hard drive or **SSD**.

Run the following command as root to make sure that rsync can access all system files and preserve the ownership:

```
# rsync -aAXv --exclude={"/dev/*","/proc/*","/sys/*","/tmp/*","/run/*","/mnt/*","/media/*","/lost+found"} / /path/to/backup/folder
```

By using the `-aAX` set of options, the files are transferred in archive mode which ensures that symbolic links, devices, permissions, ownerships, modification times, **ACLs**, and extended attributes are preserved, assuming that the target **file system** supports the feature.

The `--exclude` option causes files that match the given patterns to be excluded. The contents of `/dev`, `/proc`, `/sys`, `/tmp`, and `/run` are excluded in the above command, because they are populated at boot, although the folders themselves are *not* created. `/lost+found` is filesystem-specific. The command above depends on brace expansion available in both the **bash (https://www.gnu.org/software/bash/manual/html_node/Brace-Expansion.html)** and **zsh (http://zsh.sourceforge.net/Doc/Release/Expansion.html#Brace-Expansion)** shells. When using a different **shell**, `--exclude` patterns should be repeated manually. Quoting the exclude patterns will avoid expansion by the **shell**, which is necessary, for example, when backing up over **SSH**. Ending the excluded paths with `*` ensures that the directories themselves are created if they do not already exist.

Note:

- If you plan on backing up your system somewhere other than `/mnt` or `/media`, do not forget to add it to the list of exclude patterns to avoid an infinite loop.
- If there are any bind mounts in the system, they should be excluded as well so that the bind mounted contents is copied only once.
- If you use a **swap file**, make sure to exclude it as well.
- Consider if you want to backup the `/home/` folder. If it contains your data it might be considerably larger than the system. Otherwise consider excluding unimportant subdirectories such as `/home/*/.thumbnails/*`, `/home/*/.cache/mozilla/*`, `/home/*/.cache/chromium/*`, and `/home/*/.local/share/Trash/*`, depending on software installed on the system. If **GVFS** is installed, `/home/*/.gvfs` must be excluded to prevent rsync errors.

You may want to include additional **rsync** options, such as the following. See **rsync(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/rsync.1)** for the full list.

- If you use many hard links, consider adding the `-H` option, which is turned off by default due to its memory expense; however, it should be no problem on most modern machines. Many hard links reside under the `/usr/` directory.
- You may want to add rsync's `--delete` option if you are running this multiple times to the same backup folder. In this case make sure that the source path does not end with `/*`, or this option will only have effect on the files inside the subdirectories of the source directory, but it will have no effect on the files residing directly inside the source directory.

- If you use any sparse files, such as virtual disks, **Docker** images and similar, you should add the `-S` option.
- The `--numeric-ids` option will disable mapping of user and group names; instead, numeric group and user IDs will be transfered. This is useful when backing up over **SSH** or when using a live system to backup different system disk.
- Choosing `--info=progress2` option instead of `-v` will show the overall progress info and transfer speed instead of the list of files being transferred.

Restore a backup

If you wish to restore a backup, use the same rsync command that was executed but with the source and destination reversed.

File system cloning

rsync provides a way to do a copy of all data in a file system while preserving as much information as possible, including the file system metadata. It is a procedure of data cloning on a file system level where source and destination file systems don't need to be of the same type. It can be used for backing up, file system migration or data recovery.

rsync's *archive* mode comes close to being fit for the job, but it doesn't back up the special file system metadata such as access control lists, extended attributes or sparse file properties. For successful cloning at the file system level, some additional options need to be provided:

```
rsync -qaHAXS SOURCE_DIR DESTINATION_DIR
```

And their meaning is (from the manpage):

```
-H, --hard-links      preserve hard links
-A, --acls            preserve ACLs (implies -p)
-X, --xattrs          preserve extended attributes
-S, --sparse          handle sparse files efficiently
```

Produced copy can be simply reread and checked (for example after a data recovery attempt) at the file system level with `diff` 's recursive option:

```
diff -r SOURCE_DIR DESTINATION_DIR
```

It is possible to do a successful file system migration by using rsync as described in this article and updating the **fstab** and **bootloader** as described in **Migrate installation to new hardware**. This essentially provides a way to convert any root file system to another one.

rsync daemon

rsync can be run as daemon on a server listening on port `873` .

Edit the template `/etc/rsyncd.conf` , configure a share and **start** the `rsyncd.service` .

Note: As of rsync (https://www.archlinux.org/packages/?name=rsync)-3.1.2-5 the systemd unit `rsyncd.service` included in the package adds security feature `ProtectSystem=full` (`ProtectHome=on` has been undone in rsync (https://www.archlinux.org/packages/?name=rsync)-3.1.2-8) under the `[Service]` section. This makes the `/boot/`, `/etc/` and `/usr/` directories read-only. If you need rsyncd write system directories you can **edit** the unit and set `ProtectSystem=off` in the `[Service]` section of the overriding snippet.

Usage from client, e.g. list server content:

```
$ rsync rsync://server/share
```

transfer file from client to server:

```
$ rsync local-file rsync://server/share/
```

Consider iptables to open port `873` and user authentication.

Note: All transferred data including user authentication are not encrypted.

See also

- More usage examples can be searched in the **Community Contributions (https://bbs.archlinux.org/viewforum.php?id=27)** and **General Programming (https://bbs.archlinux.org/viewforum.php?id=33)** forums
- **Howto – local and remote snapshot backup using rsync with hard links (http://www.pointsoftware.ch/en/howto-local-and-remote-snapshot-backup-using-rsync-with-hard-links/)** Includes file deduplication with hard-links, MD5 integrity signature, 'chattr' protection, filter rules, disk quota, retention policy with exponential distribution (backups rotation while saving more recent backups than older)
- **Using SSH keys/identity files with rsync (https://stackoverflow.com/questions/5527068/how-do-you-use-an-identity-file-with-rsync)**

Retrieved from "https://wiki.archlinux.org/index.php?title=Rsync&oldid=505436"
