Learning Rsync On Rocky Linux (English version)

A book from the Documentation Team

Version : 2025/07/09

Rocky Documentation Team

Table of contents

1. Licence

RockyLinux offers Linux courseware for trainers or people wishing to learn how to administer a Linux system on their own.

RockyLinux materials are published under Creative Commons-BY-SA. This means you are free to share and transform the material, while respecting the author's rights.

BY : **Attribution**. You must cite the name of the original author.

SA : **Share Alike**.

- Creative Commons-BY-SA licence : https://creativecommons.org/licenses/by-sa/4.0/

The documents and their sources are freely downloadable from:

- https://docs.rockylinux.org
- https://github.com/rocky-linux/documentation

Our media sources are hosted at github.com. You'll find the source code repository where the version of this document was created.

From these sources, you can generate your own personalized training material using mkdocs. You will find instructions for generating your document here.

How can I contribute to the documentation project?

You'll find all the information you need to join us on our git project home page.

We wish you all a pleasant reading and hope you enjoy the content.

2. Backup Brief

What is a backup?

Backup refers to the duplication of data in the file system or database. In the event of an error or disaster, the effective data of the system can be restored in a timely manner and normal operation.

What are the backup methods?

- Full backup: refers to a one-time copy of all files, folders or data in the hard disk or database. (Pros: the best, can recover data faster. Disadvantages: take up a larger hard disk space.)
- Incremental backup: refers to the backup of the data updated after the last full backup or incremental backup. The process is like this, such as a full backup on the first day; a backup of the newly added data on the second day, as opposed to a full backup; on the third day, a backup of the newly added data on the basis of the second day, relative to the next day, and so on.
- Differential backup: Refers to the backup of the changed files after the full backup. For example, a full backup on the first day; a backup of the new data on the second day; a backup of the new data from the second day to the third day on the third day; and a backup of all the new data from the second day to the fourth day on the fourth day, and so on.
- Selective backup: Refers to backing up a part of the system.
- Cold backup: refers to the backup when the system is in shutdown or maintenance state. The backed up data is exactly the same as the data in the system during this period.
- Hot backup: Refers to the backup when the system is in normal operation. As the data in the system is updated at any time, the backed-up data has a certain lag relative to the real data of the system.
- Remote backup: refers to backing up data in another geographic location to avoid data loss and service interruption caused by fire, natural disasters, theft, etc.

2.1 rsync in brief

On a server, I backed up the first partition to the second partition, which is commonly known as "Local backup." The specific backup tools are `tar` , `dd` , `dump` , `cp` , etc. can be achieved. Although the data is backed up on this server, if the hardware fails to boot up properly, the data will not be retrieved. In order to solve this problem with the local backup, we introduced another kind of backup --- "remote backup".

Some people will say, can't I just use the `tar` or `cp` command on the first server and send it to the second server via `scp` or `sftp` ?

In a production environment, the amount of data is relatively large. First of all, `tar` or `cp` consumes a lot of time and occupies system performance. Transmission via `scp` or `sftp` also occupies a lot of network bandwidth, which is not allowed in the actual production environment. Secondly, these commands or tools need to be manually entered by the administrator and need to be combined with the crontab of the scheduled task. However, the time set by crontab is not easy to grasp, and it is not appropriate for data to be backed up if the time is too short or too long.

Therefore, there needs to be a data backup in the production environment which needs to meet the following requirements:

1. Backups transmitted over the network
2. Real-time data file synchronization
3. Less occupancy of system resources and higher efficiency

`rsync` appears to meet the above needs. It uses the GNU open source license agreement. It is a fast incremental backup tool. The latest version is 3.2.3 (2020-08-06). You can visit the Official website for more information.

In terms of platform support, most Unix-like systems are supported, whether it is GNU/Linux or BSD. In addition, there are related `rsync` under the Windows platform, such as cwRsync.

The original `rsync` was maintained by the Australian programmer Andrew Tridgell (shown in Figure 1 below), and now it has been maintained by Wayne Davison

(shown in Figure 2 below)) For maintenance, you can go to github project address to get the information you want.

note

rsync itself is only an incremental backup tool and does not have the function of real-time data synchronization (it needs to be supplemented by other programs). In addition, synchronization is one-way. If you want to realize two-way synchronization, you need to cooperate with other tools.

2.1.1 Basic Principles and Features

How does `rsync` achieve efficient one-way data synchronization backup?

The core of `rsync` is its **Checksum algorithm**. For more information, you can go to How Rsync works and The rsync algorithm. This section is beyond the author's competence and will not be covered too much.

The characteristics of `rsync` are:

- The entire directory can be updated recursively;
- Can selectively retain file synchronization attributes, such as hard link, soft link, owner, group, corresponding permissions, modification time, etc., and can retain some of the attributes;
- Support two protocols for transmission, one is ssh protocol, the other is rsync protocol

3. Preface

`rsync` needs to perform user authentication before data synchronization. **There are two protocol methods for authentication: SSH protocol and rsync protocol (the default port of rsync protocol is 873)**

- SSH protocol verification login method: use SSH protocol as the basis for user identity authentication (that is, use the system user and password of GNU/Linux itself for verification), and then perform data synchronization.
- rsync protocol verification login method: use rsync protocol for user identity authentication (non-GNU/Linux system users, similar to vsftpd virtual users), and then perform data synchronization.

Before the specific demonstration of rsync synchronization, you need to use the `rsync` command. In Rocky Linux 8, the rsync rpm package is installed by default, and the version is 3.1.3-12, as follows:

```
[root@Rocky ~]# rpm -qa|grep rsync
rsync-3.1.3-12.el8.x86_64
```

```
Basic format: rsync [options] original location target location
Commonly used options:
-a: archive mode, recursive and preserves the attributes of the file object,
which is equivalent to -rlptgoD (without -H, -A, -X)
-v: Display detailed information about the synchronization process
-z: compress when transferring files
-H: Keep hard link files
-A: retain ACL permissions
-X: retain chattr permissions
-r: Recursive mode, including all files in the directory and subdirectories
-l: still reserved for symbolic link files
-p: Permission to retain file attributes
-t: time to retain file attributes
-g: retain the group belonging to the file attribute (only for super users)
-o: retain the owner of the file attributes (only for super users)
-D: Keep device files and other special files
```

The author's personal use: `rsync -avz original location target location`

3.1 Environment Description

Item	Description
Rocky Linux 8(Server)	192.168.100.4/24
Fedora 34(client)	192.168.100.5/24

You can use Fedora 34 to upload and download

```
graph LR;
RockyLinux8-->|pull/download|Fedora34;
Fedora34-->|push/upload|RockyLinux8;
```

You can also use Rocky Linux 8 to upload and download

```
graph LR;
RockyLinux8-->|push/upload|Fedora34;
Fedora34-->|pull/download|RockyLinux8;
```

3.2 Demonstration based on SSH protocol

> **tip**
>
> Here, both Rocky Linux 8 and Fedora 34 use the root user to log in. Fedora 34 is the client and Rocky Linux 8 is the server.

3.2.1 pull/download

Since it is based on the SSH protocol, we first create a user in the server:

```
[root@Rocky ~]# useradd testrsync
[root@Rocky ~]# passwd testrsync
```

On the client side, we pull/download it, and the file on the server is /rsync/aabbcc

```
[root@fedora ~]# rsync -avz testrsync@192.168.100.4:/rsync/aabbcc /root
testrsync@192.168.100.4 ' s password:
receiving incremental file list
aabbcc
sent 43 bytes received 85 bytes 51.20 bytes/sec
total size is 0 speedup is 0.00
[root@fedora ~]# cd
```

```
[root@fedora ~]# ls
aabbcc
```

The transfer was successful.

> **tip**
>
> If the server's SSH port is not the default 22, you can specify the port in a similar way---`rsync -avz -e 'ssh -p [port]'`.

3.2.2 push/upload

```
[root@fedora ~]# touch fedora
[root@fedora ~]# rsync -avz /root/* testrsync@192.168.100.4:/rsync/
testrsync@192.168.100.4 ' s password:
sending incremental file list
anaconda-ks.cfg
fedora
rsync: mkstemp " /rsync/.anaconda-ks.cfg.KWf7JF " failed: Permission denied
(13)
rsync: mkstemp " /rsync/.fedora.fL3zPC " failed: Permission denied (13)
sent 760 bytes received 211 bytes 277.43 bytes/sec
total size is 883 speedup is 0.91
rsync error: some files/attrs were not transferred (see previous errors) (code
23) at main.c(1330) [sender = 3.2.3]
```

Prompt permission denied, how to deal with it?

First check the permissions of the /rsync/ directory. Obviously, there is no "w" permission. We can use `setfacl` to give permission:

```
[root@Rocky ~ ] # ls -ld /rsync/
drwxr-xr-x 2 root root 4096 November 2 15:05 /rsync/
```

```
[root@Rocky ~ ] # setfacl -mu:testrsync:rwx /rsync/
[root@Rocky ~ ] # getfacl /rsync/
getfacl: Removing leading ' / ' from absolute path names
# file: rsync/
# owner: root
# group: root
user::rwx
user:testrsync:rwx
group::rx
```

```
mask::rwx
other::rx
```

Try again, success!

```
[root@fedora ~ ] # rsync -avz /root/* testrsync@192.168.100.4:/rsync/
testrsync@192.168.100.4 ' s password:
sending incremental file list
anaconda-ks.cfg
fedora
sent 760 bytes received 54 bytes 180.89 bytes/sec
total size is 883 speedup is 1.08
```

4. Demonstration based on rsync protocol

In vsftpd, there are virtual users (impersonated users customized by the administrator) because it is not safe to use anonymous users and local users. We know that a server based on the SSH protocol must ensure that there is a system of users. When there are many synchronization requirements, it may be necessary to create many users. This obviously does not meet the GNU/Linux operation and maintenance standards (the more users, the more insecure), in rsync, for security reasons, there is an rsync protocol authentication login method.

How to do it?

Just write the corresponding parameters and values in the configuration file. In Rocky Linux 8, you need to manually create the file /etc/rsyncd.conf.

```
[root@Rocky ~]# touch /etc/rsyncd.conf
[root@Rocky ~]# vim /etc/rsyncd.conf
```

Some parameters and values of this file are as follows, here has more parameter descriptions:

Item	Description
address = 192.168.100.4	The IP address that rsync listens on by default
port = 873	rsync default listening port
pid file = /var/run/rsyncd.pid	File location of process pid
log file = /var/log/rsyncd.log	File location of the log
[share]	Share name
comment = rsync	Remarks or description information
path = /rsync/	The system path location where it is located
read only = yes	yes means read only, no means read and write
dont compress = *.gz *.gz2 *.zip	Which file types do not compress it
auth users = li	Enable virtual users and define what a virtual user is called. Need to create it yourself
secrets file = /etc/rsyncd_users.db	Used to specify the location of the virtual user's password file, which must end in .db. The content format of the file is "Username: Password", one per line

tip

The permission of the password file must be 600.

Write some file content to /etc/rsyncd.conf, and write the user name and password to /etc/rsyncd_users.db, the permission is 600

```
[root@Rocky ~]# cat /etc/rsyncd.conf
address = 192.168.100.4
port = 873
pid file = /var/run/rsyncd.pid
log file = /var/log/rsyncd.log
[share]
comment = rsync
path = /rsync/
read only = yes
dont compress = *.gz *.bz2 *.zip
auth users = li
secrets file = /etc/rsyncd_users.db
[root@Rocky ~]# ll /etc/rsyncd_users.db
-rw------- 1 root root 9 November 2 16:16 /etc/rsyncd_users.db
[root@Rocky ~]# cat /etc/rsyncd_users.db
li:13579
```

You may need to `dnf -y install rsync-daemon` before you can start the service: `systemctl start rsyncd.service`

```
[root@Rocky ~]# systemctl start rsyncd.service
[root@Rocky ~]# netstat -tulnp
Proto Recv-Q Send-Q Local Address           Foreign Address
State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN      691/sshd
tcp        0      0 192.168.100.4:873       0.0.0.0:*
LISTEN      4607/rsync
tcp6       0      0 :::22                   :::*
LISTEN      691/sshd
udp        0      0 127.0.0.1:323           0.
0.0.0:*                           671/chronyd
udp6       0      0 ::
1:323                :::*                                 671/chronyd
```

4.1 pull/download

Create a file in the server for verification: `[root@Rocky]# touch /rsync/rsynctest.txt`

The client does the following:

```
[root@fedora ~]# rsync -avz li@192.168.100.4::share /root
Password:
receiving incremental file list
./
rsynctest.txt
sent 52 bytes received 195 bytes 7.16 bytes/sec
total size is 883 speedup is 3.57
[root@fedora ~]# ls
aabbcc anaconda-ks.cfg fedora rsynctest.txt
```

success! In addition to the above writing based on the rsync protocol, you can also write like this: `rsync://li@10.1.2.84/share`

4.2 push/upload

```
[root@fedora ~]# touch /root/fedora.txt
[root@fedora ~]# rsync -avz /root/* li@192.168.100.4::share
Password:
sending incremental file list
rsync: [sender] read error: Connection reset by peer (104)
rsync error: error in socket IO (code 10) at io.c(784) [sender = 3.2.3]
```

You are prompted that the reading error is related to the "read only = yes" of the server . Change it to "no" and restart the service

`[root@Rocky ~]# systemctl restart rsyncd.service`

Try again, prompting you permission denied:

```
[root@fedora ~]# rsync -avz /root/* li@192.168.100.4::share
Password:
sending incremental file list
fedora.txt
rsync: mkstemp " /.fedora.txt.hxzBIQ " (in share) failed: Permission denied
(13)
sent 206 bytes received 118 bytes 92.57 bytes/sec
total size is 883 speedup is 2.73
```

```
rsync error: some files/attrs were not transferred (see previous errors) (code
23) at main.c(1330) [sender = 3.2.3]
```

Our virtual user here is li, which is mapped to the system user nobody by default. Of course, you can change it to other system users. In other words, nobody does not have write permission to the /rsync/ directory. Of course, we can use `[root@Rocky ~]# setfacl -mu:nobody:rwx /rsync/` , try again, and succeed.

```
[root@fedora ~]# rsync -avz /root/* li@192.168.100.4::share
Password:
sending incremental file list
fedora.txt
sent 206 bytes received 35 bytes 96.40 bytes/sec
total size is 883 speedup is 3.66
```

5. /etc/rsyncd.conf

In the previous article rsync demo 02 we introduced some basic parameters. This article is to supplement other parameters.

Parameters	Description
fake super = yes	yes means that you do not need the daemon to run as root to store the complete attributes of the file.
uid =	user id
gid =	Two parameters are used to specify the user and group used to transfer files when running the rsync daemon as root. The default is nobody
use chroot = yes	Whether the root directory needs to be locked before transmission, yes yes, no no. In order to increase security, rsync defaults to yes.
max connections = 4	The maximum number of connections allowed, the default value is 0, which means that there is no restriction
lock file = /var/run/rsyncd.lock	The specified lock file, which is associated with the "max connections" parameter
exclude = lost+found/	Exclude directories that do not need to be transferred
transfer logging = yes	Whether to enable ftp-like log format to record rsync uploads and downloads
timeout = 900	Specify the timeout period. If no data is transmitted within the specified time, rsync will exit directly. The unit is seconds, the default value is 0 means never time out
ignore nonreadable = yes	Whether to ignore files to which the user does not have access rights
motd file = /etc/rsyncd/rsyncd.motd	Used to specify the path of the message file. By default, there is no motd file. This message is the welcome message displayed when the user logs in.
hosts allow = 10.1.1.1/24	Used to specify which IP or network segment clients are allowed to access. You can fill in the ip, network segment, host name, host under the domain, and separate multiples with spaces. Allow everyone to access by default
hosts deny = 10.1.1.20	Which ip or network segment clients specified by the user are not allowed to access. If hosts allow and hosts deny have the same matching result, the client cannot access eventually. If the client's address is neither in the hosts allow nor in the hosts deny, the client is allowed to access. By default, there is no such parameter
auth users = li	Enable virtual users, multiple users are separated by commas in English state
syslog facility = daemon	Define the level of system log. These values can be filled in: auth, authpriv, cron, daemon, ftp, kern, lpr, mail, news, security, syslog, user, uucp, local0, local1, local2 local3, local4, local5, local6 and local7. The default value is daemon

5.1 Recommended configuration

/etc/rsyncd.conf

```
uid = nobody
gid = nobody
address = 192.168.100.4
use chroot = yes
max connections = 10
syslog facility = daemon
pid file = /var/run/rsyncd.pid
log file = /var/log/rsyncd.log
lock file = /var/run/rsyncd.lock
[file]
  comment = rsync
  path = /rsync/
  read only = no
  dont compress = *.gz *.bz2 *.zip
  auth users = li
  secrets file = /etc/rsyncd users.db
```

6. Foreword

From rsync Brief Description we know that rsync is an incremental synchronization tool. Every time the `rsync` command is executed, data can be synchronized once, but data cannot be synchronized in real time. How to do it?

With inotify-tools, this program tool can realize one-way real-time synchronization. Since it is real-time data synchronization, the prerequisite is to log in without password authentication.

Regardless of whether it is rsync protocol or SSH protocol, both can achieve password-free authentication login.

6.1 SSH protocol password-free authentication login

First, generate a public key and private key pair on the client, and keep pressing Enter after typing the command. The key pair is saved in the /root/.ssh/ directory.

```
[root@fedora ~]# ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256: TDA3tWeRhQIqzTORLaqy18nKnQOFNDhoAsNqRLo1TMg root@fedora
The key's randomart image is:
+---[RSA 2048]----+
|O+. +o+o. .+. |
|BEo oo*....o. |
|*o+o..*.. ..o |
|.+..o. = o |
|o o S |
|. o |
| o +. |
|....=. |
| .o.o. |
+----[SHA256]-----+
```

Then, use the `scp` command to upload the public key file to the server. For example, I upload this public key to the user **testrsync**

```
[root@fedora ~]# scp -P 22 /root/.ssh/id_rsa.pub root@192.168.100.4:/home/
testrsync/
```

```
[root@Rocky ~]# cat /home/testrsync/id_rsa.pub >> /home/testrsync/.ssh/
authorized_keys
```

Try to log in without secret authentication, success!

```
[root@fedora ~]# ssh -p 22 testrsync@192.168.100.4
Last login: Tue Nov 2 21:42:44 2021 from 192.168.100.5
[testrsync@Rocky ~]$
```

tip

The server configuration file **/etc/ssh/sshd_config** should be opened PubkeyAuthentication yes

6.2 rsync protocol password-free authentication login

On the client side, the rsync service prepares an environment variable for the system-**RSYNC_PASSWORD**, which is empty by default, as shown below:

```
[root@fedora ~]# echo "$RSYNC_PASSWORD"

[root@fedora ~]#
```

If you want to achieve password-free authentication login, you only need to assign a value to this variable. The value assigned is the password previously set for the virtual user li. At the same time, declare this variable as a global variable.

```
[root@Rocky ~]# cat /etc/rsyncd_users.db
li:13579
```

```
[root@fedora ~]# export RSYNC_PASSWORD=13579
```

Try it, success! No new files appear here, so the list of transferred files is not displayed.

```
[root@fedora ~]# rsync -avz li@192.168.100.4::share /root/
receiving incremental file list
./

sent 30 bytes received 193 bytes 148.67 bytes/sec
total size is 883 speedup is 3.96
```

tip

You can write this variable into **/etc/profile** to make it take effect permanently. The content is: `export RSYNC_PASSWORD=13579`

7. Compile and install

Perform the following operations in the server. In your environment, some dependent packages may be missing. Install them by using:

`dnf -y install autoconf automake libtool`

```
[root@Rocky ~]# wget -c https://github.com/inotify-tools/inotify-tools/archive/
refs/tags/3.21.9.6.tar.gz
[root@Rocky ~]# tar -zvxf 3.21.9.6.tar.gz -C /usr/local/src/
[root@Rocky ~]# cd /usr/local/src/inotify-tools-3.21.9.6/
[root@Rocky /usr/local/src/inotify-tools-3.21.9.6]# ./autogen.sh && \
./configure --prefix=/usr/local/inotify-tools && \
make && \
make install
...
[root@Rocky ~]# ls /usr/local/inotify-tools/bin/
inotifywait inotifywatch
```

Append the environment variable PATH, write it to the configuration file and let it take effect permanently.

```
[root@Rocky ~]# vim /etc/profile
...
PATH=$PATH:/usr/local/inotify-tools/bin/
[root@Rocky ~]# . /etc/profile
```

Why not use the inotify-tools RPM package of the EPEL repository? And the way to use source code to compile and install?

The author personally believes that remote data transmission is a matter of efficiency, especially in a production environment, where there are a large number of files to be synchronized and a single file is particularly large. In addition, the new version will have some bug fixes and function expansions, and perhaps the transmission efficiency of the new version will be higher, so I recommend installing inotify-tools by source code. Of course, this is the author's personal suggestion, not every user must follow.

7.1 Kernel parameter adjustment

You can adjust the kernel parameters according to the needs of the production environment. By default, there are three files in **/proc/sys/fs/inotity/**

```
[root@Rocky ~]# cd /proc/sys/fs/inotify/
[root@Rocky /proc/sys/fs/inotify]# cat max_queued_events ;cat
max_user_instances ;cat max_user_watches
16384
128
28014
```

- max_queued_events-maximum monitor queue size, default 16384
- max_user_instances-the maximum number of monitoring instances, the default is 128
- max_user_watches-the maximum number of files monitored per instance, the default is 8192

Write some parameters and values to **/etc/sysctl.conf**, examples are as follows. Then use `sysctl -p` to make the files take effect

```
fs.inotify.max_queued_events = 16384
fs.inotify.max_user_instances = 1024
fs.inotify.max_user_watches = 1048576
```

7.2 Related commands

The inotify-tools tool has two commands, namely:

- **inotifywait**: for continuous monitoring, real-time output results. It is generally used with the rsync incremental backup tool. Because it is a file system monitoring, it can be used with a script. We will introduce the specific script writing later.
- **inotifywatch**: for short-term monitoring, output results after the task is completed.

`inotifywait` mainly has the following options:

```
-m means continuous monitoring
-r Recursive monitoring
-q Simplify output information
-e specifies the event type of monitoring data, multiple event types are
separated by commas in English status
```

The event types are as follows:

Event Type	Description
access	Access to the contents of a file or directory
modify	The contents of the file or directory are written
attrib	The attributes of the file or directory are modified
close_write	File or directory is opened in writable mode and then closed
close_nowrite	File or directory is closed after being opened in read-only mode
close	Regardless of the read/write mode, the file or directory is closed
open	File or directory is opened
moved_to	A file or directory is moved to the monitored directory
moved_from	A file or directory is moved from the monitored directory
move	There are files or directories that are moved to or removed from the monitoring directory
move_self	The monitored file or directory has been moved
create	There are files or directories created in the monitored directory
delete	A file or directory in the monitored directory is deleted
delete_self	The file or directory has been deleted
unmount	File system containing unmounted files or directories

Example: `[root@Rocky ~]# inotifywait -mrq -e create,delete /rsync/`

7.3 Demonstration of `inotifywait` command

Type the command in the first terminal pts/0, and the window is locked after pressing Enter, indicating that it is monitoring

```
[root@Rocky ~]# inotifywait -mrq -e create,delete /rsync/
```

In the second terminal pts/1, go to the /rsync/ directory and create a file.

```
[root@Rocky ~]# cd /rsync/
[root@Rocky /rsync]# touch inotify
```

Back to the first terminal pts/0, the output information is as follows:

```
[root@Rocky ~]# inotifywait -mrq -e create,delete /rsync/
/rsync/ CREATE inotify
```

7.4 Combination of `inotifywait` and `rsync`

> **tip**
>
> We are operating in Rocky Linux 8 server, using SSH protocol for demonstration.

For the password-free authentication login of the SSH protocol, please refer to rsync password-free authentication login, which is not described here. An example of the content of a bash script is as follows. You can add different options after the command according to your needs to meet your needs. For example, you can also add `--delete` after the `rsync` command.

```
#!/bin/bash
a="/usr/local/inotify-tools/bin/inotifywait -mrq -e modify,move,create,delete /
rsync/"
b="/usr/bin/rsync -avz /rsync/* testfedora@192.168.100.5:/home/testfedora/"
$a | while read directory event file
    do
        $b &>> /tmp/rsync.log
    done
```

```
[root@Rocky ~]# chmod +x rsync_inotify.sh
[root@Rocky ~]# bash /root/rsync_inotify.sh &
```

> **tip**
>
> When using the SSH protocol for data synchronization transmission, if the SSH service port of the target machine is not 22, you can use a method similar to this—— `b="/usr/bin/rsync -avz -e 'ssh -p [port-number]' /rsync/* testfedora@192.168.100.5:/home/testfedora/"`

tip

If you want to start this script at boot `[root@Rocky ~]# echo "bash /root/rsync_inotify.sh &" >> /etc/rc.local` `[root@Rocky ~]# chmod +x /etc/rc.local`

If you are using the rsync protocol for synchronization, you need to configure the rsync service of the target machine, please refer to rsync demo 02, rsync configuration file, rsync free Secret authentication login

8. Brief

As we mentioned earlier, one-way synchronization uses rsync + inotify-tools. In some special usage scenarios, two-way synchronization may be required, which requires inotify-tools + unison.

8.1 Environment preparation

- Both Rocky Linux 8 and Fedora 34 require source code compilation and installation **inotify-tools**, which is not specifically expanded here.
- Both machines must be password-free login authentication, here we use the SSH protocol for
- ocaml uses v4.12.0, unison uses v2.51.4.

After the environment is ready, it can be verified:

```
[root@Rocky ~]# inotifywa
inotifywait inotifywatch
[root@Rocky ~]# ssh -p 22 testrsync@192.168.100.5
Last login: Thu Nov 4 13:13:42 2021 from 192.168.100.4
[testrsync@fedora ~]$
```

```
[root@fedora ~]# inotifywa
inotifywait inotifywatch
[root@fedora ~]# ssh -p 22 testrsync@192.168.100.4
Last login: Wed Nov 3 22:07:18 2021 from 192.168.100.5
[testrsync@Rocky ~]$
```

tip

The configuration files of the two machines **/etc/ssh/sshd_config** should be opened PubkeyAuthentication yes

8.2 Rocky Linux 8 install unison

Ocaml is a programming language, and the bottom layer of unison depends on it.

```
[root@Rocky ~]# wget -c https://github.com/ocaml/ocaml/archive/refs/tags/
4.12.0.tar.gz
```

```
[root@Rocky ~]# tar -zvxf 4.12.0.tar.gz -C /usr/local/src/
[root@Rocky ~]# cd /usr/local/src/ocaml-4.12.0
[root@Rocky /usr/local/src/
ocaml-4.12.0]# ./configure --prefix=/usr/local/ocaml && make world opt && make
install
...
[root@Rocky ~]# ls /usr/local/ocaml/
bin lib man
[root@Rocky ~]# echo PATH=$PATH:/usr/local/ocaml/bin >> /etc/profile
[root@Rocky ~]# . /etc/profile
```

```
[root@Rocky ~]# wget -c https://github.com/bcpierce00/unison/archive/refs/tags/
v2.51.4.tar.gz
[root@Rocky ~]# tar -zvxf v2.51.4.tar.gz -C /usr/local/src/
[root@Rocky ~]# cd /usr/local/src/unison-2.51.4/
[root@Rocky /usr/local/src/unison-2.51.4]# make UISTYLE=txt
...
[root@Rocky /usr/local/src/unison-2.51.4]# ls src/unison
src/unison
[root@Rocky /usr/local/src/unison-2.51.4] cp -p src/unison /usr/local/bin
```

8.3 Fedora 34 install unison

The same operation.

```
[root@fedora ~]# wget -c https://github.com/ocaml/ocaml/archive/refs/tags/
4.12.0.tar.gz
[root@feodora ~]# tar -zvxf 4.12.0.tar.gz -C /usr/local/src/
[root@fedora ~]# cd /usr/local/src/ocaml-4.12.0
[root@fedora /usr/local/src/ocaml-4.12.0]# ./configure --prefix=/usr/local/
ocaml && make world opt && make install
...
[root@fedora ~]# ls /usr/local/ocaml/
bin lib man
[root@fedora ~]# echo PATH=$PATH:/usr/local/ocaml/bin >> /etc/profile
[root@fedora ~]#. /etc/profile
```

```
[root@fedora ~]# wget -c https://github.com/bcpierce00/unison/archive/refs/
tags/v2.51.4.tar.gz
[root@fedora ~]# tar -zvxf v2.51.4.tar.gz -C /usr/local/src/
[root@fedora ~]# cd /usr/local/src/unison-2.51.4/
[root@fedora /usr/local/src/unison-2.51.4]# make UISTYLE=txt
...
[root@fedora /usr/local/src/unison-2.51.4]# ls src/unison
```

```
src/unison
[root@fedora /usr/local/src/unison-2.51.4]# cp -p src/unison /usr/local/bin
```

8.4 Demo

Our requirement is-Rocky Linux 8's /dir1/ directory is automatically synchronized to Fedora 34's /dir2/ directory; at the same time, Fedora 34's /dir2/ directory is automatically synchronized to Rocky Linux 8's /dir1/ directory

8.4.1 Configure Rocky Linux 8

```
[root@Rocky ~]# mkdir /dir1
[root@Rocky ~]# setfacl -m u:testrsync:rwx /dir1/
[root@Rocky ~]# vim /root/unison1.sh
#!/bin/bash
a="/usr/local/inotify-tools/bin/inotifywait -mrq -e create,delete,modify,move /
dir1/"
b="/usr/local/bin/unison -batch /dir1/ ssh://testrsync@192.168.100.5//dir2"
$a | while read directory event file
do
    $b &>> /tmp/unison1.log
done
[root@Rocky ~]# chmod +x /root/unison1.sh
[root@Rocky ~]# bash /root/unison1.sh &
[root@Rocky ~]# jobs -l
```

8.4.2 Configure Fedora 34

```
[root@fedora ~]# mkdir /dir2
[root@fedora ~]# setfacl -m u:testrsync:rwx /dir2/
[root@fedora ~]# vim /root/unison2.sh
#!/bin/bash
a="/usr/local/inotify-tools/bin/inotifywait -mrq -e create,delete,modify,move /
dir2/"
b="/usr/local/bin/unison -batch /dir2/ ssh://testrsync@192.168.100.4//dir1"
$a | while read directory event file
do
    $b &>> /tmp/unison2.log
done
[root@fedora ~]# chmod +x /root/unison2.sh
```

```
[root@fedora ~]# bash /root/unison2.sh &
[root@fedora ~]# jobs -l
```

tip

For two-way synchronization, the scripts of both machines must be started, otherwise an error will be reported.

tip

If you want to start this script at boot `[root@Rocky ~]# echo "bash /root/unison1.sh &" >> /etc/rc.local`
`[root@Rocky ~]# chmod +x /etc/rc.local`

tip

If you want to stop the corresponding process of this script, you can find it in the `htop` command and then **kill**