Linux常用基础工具

- Mosh 基础使用指南
- Supervisor 在 Rocky Linux 上的安装与使用指南
- Rsyncd 在 Rocky Linux 上的安装与使用指南
- SSH 使用指南
- cURL 常用操作指南

Mosh 基础使用指南

Mosh（ Mobile Shell）是一个比传统 SSH 更强大的远程终端工具，特别适合移动和不稳定网络环境下的使用。

一、 Mosh 简介

Mosh 的主要优势:

- 在网络切换或中断时保持连接
- 输入即时响应（无延迟显示）
- 预测性本地回显
- 更安全的连接方式

二、安装 Mosh

1. Linux 系统安装

```
# Ubuntu/Debian
sudo apt-get install mosh

# CentOS/RHEL
sudo yum install mosh

# Fedora
sudo dnf install mosh
```

2. macOS 安装

```
brew install mosh
```

3. Windows 使用

Windows 用户需要:

1. 安装 WSL 或 Cygwin
2. 在其中安装 mosh
3. 使用支持 mosh 的终端如 Windows Terminal

三、基本使用方法

1. 简单连接

```
mosh username@remote_host
```

2. 指定端口

```
mosh --ssh="ssh -p 2222" username@remote_host
```

3. 使用特定 SSH 密钥

```
mosh --ssh="ssh -i ~/.ssh/id_rsa" username@remote_host
```

四、常用参数

参数	说明
`-p PORT`	指定 UDP 端口范围（如 60000:61000）
`--predict=never`	禁用预测性回显
`--predict=always`	总是使用预测性回显
`--predict=adaptive`	自适应预测（默认）
`--ssh="COMMAND"`	指定 SSH 连接命令
`-a`	强制使用本地 IP 地址
`-n`	不初始化 SSH 会话（用于调试）

五、高级用法

1. 通过跳板机连接

```
mosh --ssh="ssh -J jump_user@jump_host" user@target_host
```

2. 保持连接的后台会话

```
# 使用 tmux 或 screen 配合 mosh
mosh user@host -- tmux new -A -s session_name
```

3. 指定服务器上的 mosh-server 路径

```
mosh --server="/usr/local/bin/mosh-server" user@host
```

六、Mosh 与 SSH 的区别

1. 协议差异:
   - SSH 使用 TCP
   - Mosh 使用 UDP (默认端口 60000-61000)
2. 连接恢复:
   - SSH 断连后需要重新连接
   - Mosh 在网络恢复后自动重连
3. 输入响应:
   - SSH 有输入延迟
   - Mosh 提供即时本地回显

七、防火墙配置

Mosh 需要 UDP 端口开放:

```
# 开放 UDP 60000-61000 端口
sudo ufw allow 60000:61000/udp
```

八、常见问题解决

1. 连接失败

```
# 确保服务器安装了mosh
ssh user@host "which mosh-server"

# 检查UDP端口是否开放
telnet host 60000
```

2. 解决乱码问题

```
# 指定UTF-8编码
mosh --ssh="ssh -o SendEnv=LC_*" user@host
```

3. 提高连接速度

```
# 使用压缩
mosh --ssh="ssh -C" user@host
```

九、退出Mosh会话

退出方式与SSH相同:

1. 输入 `exit` 或 `logout`
2. 或者按 `Ctrl-D`

十、Mosh客户端选项

查看所有选项:

```
mosh --help
```

Mosh 是替代 SSH 的优秀工具，特别适合网络不稳定环境下的远程工作。虽然配置略复杂，但提供的稳定性和用户体验提升值得投入时间学习。

Supervisor 在 Rocky Linux 上的安装与使用指南

Supervisor 是一个用 Python 编写的进程控制系统，允许用户监控和控制类 UNIX 系统上的多个进程。下面是在 Rocky Linux 上安装和配置 Supervisor 的完整指南。

一、安装 Supervisor

1. 安装 EPEL 仓库（如果尚未安装）

```
sudo dnf install epel-release
sudo dnf update
```

2. 安装 Supervisor

```
sudo dnf install supervisor
```

3. 启动并设置开机自启

```
sudo systemctl enable supervisord
sudo systemctl start supervisord
```

4. 验证安装

```
sudo supervisorctl status
```

二、基本配置

1. 主配置文件位置

主配置文件位于 `/etc/supervisord.conf`

2. 包含自定义配置

默认配置会包含 `/etc/supervisord.d/` 目录下的所有 `.ini` 文件:

```
[include]
files = /etc/supervisord.d/*.ini
```

三、配置管理进程

1. 创建自定义进程配置

在 `/etc/supervisord.d/` 中创建新的配置文件，例如 `myapp.ini`:

```
[program:myapp]
command=/usr/bin/python3 /path/to/your/app.py
directory=/path/to/your/app
user=yourusername
autostart=true
autorestart=true
startretries=3
stderr_logfile=/var/log/myapp.err.log
stdout_logfile=/var/log/myapp.out.log
environment=PYTHONPATH="/path/to/your/app"
```

2. 常用配置选项

选项	说明
`command`	要执行的命令
`directory`	运行命令前切换到的目录
`user`	运行进程的用户

选项	说明
`autostart`	是否随 Supervisor 启动而自动启动
`autorestart`	是否自动重启 (true/false/unexpected)
`startretries`	启动失败后的重试次数
`stderr_logfile`	标准错误日志文件
`stdout_logfile`	标准输出日志文件
`environment`	设置环境变量

四、 Supervisor 命令使用

1. 重新加载配置

```
sudo supervisorctl reread
sudo supervisorctl update
```

2. 管理进程

```
# 启动进程
sudo supervisorctl start myapp

# 停止进程
sudo supervisorctl stop myapp

# 重启进程
sudo supervisorctl restart myapp

# 查看所有进程状态
sudo supervisorctl status

# 查看特定进程状态
sudo supervisorctl status myapp
```

3. 查看日志

```
# 查看主日志
sudo tail -f /var/log/supervisor/supervisord.log

# 查看应用日志
sudo tail -f /var/log/myapp.out.log
```

五、 Web 界面 (可选)

1. 启用 Web 界面

编辑 `/etc/supervisord.conf`，取消注释或添加以下内容:

```
[inet_http_server]
port=*:9001
username=admin
password=yourpassword
```

2. 重启 Supervisor

```
sudo systemctl restart supervisord
```

3. 访问 Web 界面

通过浏览器访问 `http://your-server-ip:9001`

六、高级配置

1. 进程组管理

```
[group:mygroup]
programs=myapp1,myapp2
priority=999
```

2. 事件监听

```
[eventlistener:memmon]
command=memmon -p myapp=200MB
events=TICK_60
```

3. 日志轮转

安装 `logrotate` 并创建 `/etc/logrotate.d/supervisor`:

```
/var/log/supervisor/*.log {
    daily
    missingok
    rotate 14
    compress
    delaycompress
    notifempty
    create 0640 root root
    postrotate
        /usr/bin/supervisorctl pid > /dev/null && /usr/bin/supervisorctl reopen_logs
    endscript
}
```

七、故障排除

1. 常见问题解决

- 配置更改不生效 :

  ```
  sudo supervisorctl reread
  sudo supervisorctl update
  ```

- 进程无法启动 : 检查 `/var/log/supervisor/supervisord.log` 和应用的错误日志
- 权限问题 : 确保配置中指定的用户有执行命令和写入日志的权限

2. 日志位置

- 主日志：`/var/log/supervisor/supervisord.log`
- 子进程日志：在各自配置中指定的位置

八、安全注意事项

1. 不要以 root 用户运行应用程序
2. Web 界面应设置强密码
3. 考虑限制 Web 界面的访问 IP
4. 定期检查日志文件
5. 保持 Supervisor 更新到最新版本

通过以上步骤，您可以在 Rocky Linux 上成功安装和配置 Supervisor，实现对关键进程的有效管理和监控。

Rsyncd 在 Rocky Linux 上的安装与使用指南

Rsyncd 是 rsync 的守护进程模式，允许将 rsync 作为服务运行，实现持续的文件同步服务。以下是在 Rocky Linux 8/9 上安装和配置 rsyncd 的完整指南。

一、安装 rsyncd

1. 安装 rsync 软件包

```
sudo dnf install rsync
```

2. 检查版本确认安装成功

```
rsync --version
```

二、配置 rsyncd 服务

1. 创建主配置文件

```
sudo vi /etc/rsyncd.conf
```

2. 基本配置示例

```
# 全局配置
uid = nobody
gid = nobody
```

```
use chroot = yes
max connections = 4
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
log file = /var/log/rsyncd.log
timeout = 300

# 模块配置
[backup]
    path = /data/backup
    comment = Backup Directory
    read only = no
    list = yes
    auth users = rsyncuser
    secrets file = /etc/rsyncd.secrets
    hosts allow = 192.168.1.0/24
```

3. 创建认证文件

```
sudo vi /etc/rsyncd.secrets
```

内容格式:

```
用户名     :密码
```

例如:

```
rsyncuser:mypassword123
```

设置文件权限:

```
sudo chmod 600 /etc/rsyncd.secrets
sudo chown root:root /etc/rsyncd.secrets
```

4. 创建共享目录

```
sudo mkdir -p /data/backup
sudo chown nobody:nobody /data/backup
```

三、启动和管理 rsyncd 服务

1. 启用并启动服务

```
sudo systemctl enable rsyncd
sudo systemctl start rsyncd
```

2. 检查服务状态

```
sudo systemctl status rsyncd
```

3. 防火墙配置

```
# 开放 873端口
sudo firewall-cmd --add-port=873/tcp --permanent
sudo firewall-cmd --reload
```

4. SELinux 配置（如启用）

```
sudo setsebool -P rsync_full_access=1
```

四、客户端连接使用

1. 从客户端同步到服务器（推送）

```
rsync -avz /local/path/ rsyncuser@server::backup
```

2. 从服务器同步到客户端（拉取）

```
rsync -avz rsyncuser@server::backup /local/path/
```

3. 使用密码文件（避免交互输入）

创建客户端密码文件 `~/rsync.pass`：

```
mypassword123
```

设置权限：

```
chmod 600 ~/rsync.pass
```

使用密码文件同步：

```
rsync -avz --password-file=~/rsync.pass /local/path/ rsyncuser@server::backup
```

五、高级配置选项

1. 模块配置参数

参数	说明
`path`	模块对应的文件系统路径
`comment`	模块描述信息
`read only`	是否只读（yes/no）
`list`	是否允许列出模块内容（yes/no）
`auth users`	允许访问的用户名（逗号分隔）
`secrets file`	认证文件路径
`hosts allow`	允许访问的客户端 IP
`hosts deny`	拒绝访问的客户端 IP
`exclude`	排除的文件/目录
`include`	包含的文件/目录

2. 日志轮转配置

创建 `/etc/logrotate.d/rsyncd`：

```
/var/log/rsyncd.log {
    missingok
    notifempty
    sharedscripts
    delaycompress
    postrotate
        /bin/kill -HUP `cat /var/run/rsyncd.pid 2>/dev/null` 2>/dev/null || true
    endscript
}
```

六、安全最佳实践

1. 使用非默认端口：

   ```
   port = 8873
   ```

   并相应调整防火墙规则
2. 限制访问 **IP**：

   ```
   hosts allow = 192.168.1.100, 10.0.0.0/24
   ```

3. 使用 **SSH** 隧道（更安全）：

   ```
   rsync -avz -e "ssh -p 22" /local/path/ user@server:/remote/path/
   ```

4. 定期更换密码
5. 监控日志文件：

   ```
   sudo tail -f /var/log/rsyncd.log
   ```

七、故障排除

1. 连接被拒绝：
   - 检查服务是否运行：`systemctl status rsyncd`
   - 检查防火墙设置
   - 验证端口是否监听：`ss -tulnp | grep rsync`
2. 认证失败：
   - 检查 `/etc/rsyncd.secrets` 文件权限
   - 确认用户名密码正确
   - 检查 SELinux 状态

3. 权限问题：
    - 确认共享目录权限
    - 检查 uid/gid设置

通过以上配置，您可以在 Rocky Linux上建立稳定安全的 rsync 文件同步服务。根据实际需求调整配置参数，并定期检查日志和维护系统安全。

SSH使用指南

什么是SSH

SSH (Secure Shell)是一种加密的网络协议，用于安全地远程登录和管理系统。它替代了不安全的Telnet和rlogin等协议。

基本用法

1. 连接到远程服务器

```
ssh username@hostname
```

示例：

```
ssh user@example.com
```

2. 指定端口连接（默认22）

```
ssh -p port_number username@hostname
```

示例：

```
ssh -p 2222 user@example.com
```

密钥认证

1. 生成SSH密钥对

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

- `-t` 指定密钥类型（rsa, ed25519 等）
- `-b` 指定密钥长度
- `-C` 添加注释

2. 将公钥复制到远程服务器

```
ssh-copy-id username@hostname
```

或手动复制 `~/.ssh/id_rsa.pub` 内容到远程服务器的 `~/.ssh/authorized_keys` 文件中

配置文件

编辑 `~/.ssh/config` 可以简化连接:

```
Host myserver
    HostName example.com
    User username
    Port 2222
    IdentityFile ~/.ssh/id_rsa
```

配置后只需使用:

```
ssh myserver
```

常用选项

选项	说明
`-v`	详细模式（可叠加 -vv, -vvv）
`-X`	启用 X11 转发
`-L`	本地端口转发
`-R`	远程端口转发
`-D`	动态端口转发（SOCKS 代理）

选项	说明
`-N`	不执行远程命令
`-f`	后台运行

端口转发示例

1. 本地端口转发

```
ssh -L local_port:remote_host:remote_port username@hostname
```

2. 远程端口转发

```
ssh -R remote_port:local_host:local_port username@hostname
```

文件传输

1. 使用 scp 复制文件

从本地到远程:

```
scp local_file username@hostname:remote_path
```

从远程到本地:

```
scp username@hostname:remote_file local_path
```

2. 使用 sftp 交互式传输

```
sftp username@hostname
```

安全建议

1. 禁用 root 登录 (`PermitRootLogin no` in `/etc/ssh/sshd_config`)
2. 使用密钥认证而非密码
3. 更改默认 SSH 端口
4. 限制可登录用户 (`AllowUsers` in sshd_config)
5. 使用 fail2ban 防止暴力破解

故障排查

1. 检查服务状态:

```
systemctl status sshd
```

2. 检查日志:

```
journalctl -u sshd
```

3. 测试连接:

```
ssh -v username@hostname
```

cURL 常用操作指南

cURL（`Client URL`）是一个强大的命令行工具，用于传输数据，支持多种协议（HTTP/HTTPS/FTP/SFTP 等）。它广泛用于 API 测试、文件传输和网络调试。

---

1. 基本请求

GET 请求

```
curl https://example.com
```

- 获取网页内容
- 带参数：

  ```
  curl "https://example.com/api?param1=value1&param2=value2"
  ```

POST 请求

```
curl -X POST https://example.com/api \
     -d "key1=value1&key2=value2"
```

- 发送 **JSON** 数据：

  ```
  curl -X POST https://example.com/api \
       -H "Content-Type: application/json" \
       -d '{"key1":"value1", "key2":"value2"}'
  ```

---

2. 请求头设置

添加自定义 Header

```
curl -H "Authorization: Bearer token123" \
    -H "User-Agent: MyApp/1.0" \
    https://example.com/api
```

- 查看服务器返回的 **Headers**:

  ```
  curl -I https://example.com  # 只显示 Headers
  ```

---

3. 文件上传 & 下载

上传文件

```
curl -X POST https://example.com/upload \
    -F "file=@/path/to/file.txt" \
    -F "name=myfile"
```

- `-F` 用于 `multipart/form-data` 上传

下载文件

```
curl -O https://example.com/file.zip  # 保存为原文件名
curl -o custom_name.zip https://example.com/file.zip  # 自定义文件名
```

---

4. 认证 & Cookies

Basic Auth

```
curl -u username:password https://example.com
```

- 或只输入用户名，交互式输入密码 ：

```
curl -u username https://example.com
```

Cookie 管理

```
curl --cookie "name=value" https://example.com
```

- 保存 **Cookie** 到文件 ：

```
curl -c cookies.txt https://example.com/login
```

- 使用 **Cookie** 文件：

```
curl -b cookies.txt https://example.com/dashboard
```

---

5. 调试 & 高级选项

显示详细请求信息

```
curl -v https://example.com  # 显示请求 & 响应 Headers
curl --trace-ascii debug.txt https://example.com  # 输出调试信息到文件
```

跟随重定向

```
curl -L https://example.com  # 自动跳转
```

限速下载

```
curl --limit-rate 100K -O https://example.com/largefile.zip
```

- `100K` = 100KB/s，可设置 `M` (MB/s) 或 `G` (GB/s)

---

6. 代理设置

HTTP/HTTPS 代理

```
curl -x http://proxy-server:8080 https://example.com
```

- 带认证的代理 :

  ```
  curl -x http://user:pass@proxy-server:8080 https://example.com
  ```

SOCKS5 代理

```
curl --socks5 127.0.0.1:1080 https://example.com
```

---

7. 保存响应到变量（ Bash）

```
response=$(curl -s https://example.com/api)
echo "$response"
```

- `-s` 静默模式（不显示进度）

---

8. 常用组合命令

检查 API 返回状态码

```
curl -s -o /dev/null -w "%{http_code}" https://example.com/api
```

- `-o /dev/null` 丢弃响应体
- `-w "%{http_code}"` 只显示 HTTP 状态码

测试 **API** 响应时间

```
curl -w "DNS: %{time_namelookup} | Connect: %{time_connect} | Total: %{time_total}\n" \
    -o /dev/null -s https://example.com
```

总结

用途	命令示例
GET 请求	`curl https://example.com`
POST 请求	`curl -X POST -d "data" https://example.com/api`
JSON 请求	`curl -H "Content-Type: application/json" -d '{"key":"value"}'`
下载文件	`curl -O https://example.com/file.zip`
上传文件	`curl -F "file=@localfile.txt" https://example.com/upload`
认证	`curl -u user:pass https://example.com`
代理	`curl -x http://proxy:8080 https://example.com`
调试	`curl -v https://example.com`

cURL 功能非常强大，适用于自动化脚本、API 测试和网络调试。更多选项可查看 `man curl` 或 `curl --help`。