连接远程服务器并通过ssh协议远程执行命令

前言介绍

1、`paramiko` 库是一个用于做远程控制的模块，使用该模块可以对远程服务器进行命令或文件操作。

2、`paramiko` 库是用 python 语言写的一个模块，遵循 SSH2 协议，支持以加密和认证的方式，进行远程服务器的连接。 `paramiko` 库支持 Linux，Solaris，BSD，MacOS X，Windows 等平台通过 SSH 从一个平台连接到另外一个平台。

3、利用 `paramiko` 模块，可以方便的进行 **ssh 连接**和 sftp 协议进行 **sftp 文件传输**。

4、`paramiko` 模块包含了两个核心组件： `SSHClient` 和 `SFTPClient` 。

- `SSHClient` 类：`SSHClient` 类是**与 SSH 服务器会话**的高级表示。该类集成了 `Transport`，`Channel` 和 `SFTPClient` 类。
- `SFTPClient` 类：该类通过一个打开的 SSH Transport 会话**创建 SFTP 会话通道并执行远程文件操作**。

5、名词介绍：

```
Channel: 是一种类Socket，一种安全的SSH传输通道;
Transport: 是一种加密的会话（但是这样一个对象的Session并未建立），并且创建了一个加密的tunnels，这个tunnels叫做Channel;
Channel: 是一种类Socket，一种安全的SSH传输通道;
Session: 是client与Server保持连接的对象，用connect()/start_client()/start_server()开始会话;
hostname(str类型)，连接的目标主机地址;
port(int类型)，连接目标主机的端口，默认为22;
username(str类型)，校验的用户名(默认为当前的本地用户名);
password(str类型)，密码用于身份校验或解锁私钥;
pkey(Pkey类型)，私钥方式用于身份验证;
key_filename(str or list(str)类型)，一个文件名或文件名列表，用于私钥的身份验证;
timeout(float类型)，一个可选的超时时间(以秒为单位)的TCP连接;
allow_agent(bool类型)，设置为False时用于禁用连接到SSH代理;
look_for_keys(bool类型)，设置为False时用于来禁用在~/.ssh中搜索私钥文件;
compress(bool类型)，设置为True时打开压缩。
```

下载安装

```
pip3 install paramiko
```

paramiko 库中 SSHClient 模块的使用

1、作用：**用于连接远程服务器并执行基本命令**。

2、远程连接分为两种：（1）基于用户名密码连接远程服务器 （2）基于公钥秘钥连接远程服务器。

3、通过使用 `paramiko` 库远程操作服务器，其实本质也分为两种：（1）使用 `SSHClient` 类（2） `SSHClient` 类封装 `Transport` 类

基于用户名密码的连接

远程执行命令：

```
exec_command(command, bufsize=-1, timeout=None, get_pty=False,
environment=None)
```

1、参数说明：

- `command` （str类型），执行的命令串；
- `bufsize` （int类型），文件缓冲区大小，默认为-1（不限制）

2、使用 `exec_command` 方法执行命令会返回三个信息：

- 标准输入内容（用于实现交互式命令）--- `stdin`
- 标准输出（保存命令的正常执行结果）--- `stdout`
- 标准错误输出（保存命令的错误信息）--- `stderr`

3、通过 `exec_command` 方法命令执行完毕后，通道将关闭，不能再使用。如果想继续执行另一个命令，必须打开一个新通道。

ssh对象连接远程服务器并执行命令获取控制台打印的信息：

```python
import paramiko

# 创建SSH对象（ssh_clint）
ssh_clint = paramiko.SSHClient()

# 通过这个set_missing_host_key_policy方法用于实现登录是需要确认输入yes，否则保存
ssh_clint.set_missing_host_key_policy(paramiko.AutoAddPolicy())

# 使用connect类来连接远程服务器
ssh_clint.connect(hostname='172.16.1.1', port=22, username='test',
password='123')

# 使用exec_command方法执行命令，并使用变量接收命令的返回值并用print输出
stdin, stdout, stderr = ssh_clint.exec_command('df')

# 获取命令结果
result = stdout.read()
print(result.decode('utf-8'))

# 关闭连接
ssh_clint.close()
```

运行结果：

```
/usr/local/python3/bin/python3.9 /home/test/PycharmProjects/hls_practice/27.py
文件系统           1K-块     已用       可用 已用% 挂载点
udev             8027012        0   8027012    0% /dev
tmpfs            1612488     2116   1610372    1% /run
/dev/nvme0n1p2 490617784 32841660 432780644    8% /
tmpfs            8062420   216216   7846204    3% /dev/shm
tmpfs               5120        4      5116    1% /run/lock
tmpfs            8062420        0   8062420    0% /sys/fs/cgroup
/dev/loop0         56960    56960         0  100% /snap/core18/2409
/dev/loop1           128      128         0  100% /snap/bare/5
/dev/loop2        168832   168832         0  100% /snap/gnome-3-28-1804/161
/dev/loop3         63488    63488         0  100% /snap/core20/1581
/dev/loop4        224256   224256         0  100% /snap/gnome-3-34-1804/77
/dev/loop5        261760   261760         0  100% /snap/gnome-3-34-1804/36
/dev/loop7         66944    66944         0  100% /snap/dingtalk-notifier/8
/dev/loop6        410496   410496         0  100% /snap/gnome-3-38-2004/112
/dev/loop15        93952    93952         0  100% /snap/gtk-common-themes/1535
/dev/loop10        63488    63488         0  100% /snap/core20/1587
/dev/loop16        48128    48128         0  100% /snap/snapd/16292
/dev/loop13        51072    51072         0  100% /snap/snap-store/467
/dev/loop8         55552    55552         0  100% /snap/snap-store/558
/dev/loop14        48128    48128         0  100% /snap/snapd/16010
/dev/loop11        83328    83328         0  100% /snap/gtk-common-themes/1534
/dev/loop9        260224   260224         0  100% /snap/gnome-3-38-2004/106
/dev/loop12        56960    56960         0  100% /snap/core18/2538
/dev/nvme0n1p1    523248     5356    517892    2% /boot/efi
/dev/loop17       177152   177152         0  100% /run/wine
tmpfs            1612484     1988   1610496    1% /run/user/1000
```

使用try-except捕获异常：

```python
import paramiko
import sys

# 定义函数ssh,把操作内容写到函数里
def sshExeCMD():
    # 定义一个变量ssh_clint使用SSHClient类用来后边调用
    ssh_client = paramiko.SSHClient()
    # 通过这个set_missing_host_key_policy方法用于实现登录是需要确认输入yes，否则保存
    ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    # 使用try做异常捕获
    try:
        # 使用connect类来连接服务器
        ssh_client.connect(hostname="192.168.1.110", port=22, username="test", password="123")
    # 如果上边命令报错吧报错信息定义到err变量，并输出。
    except Exception as err:
        print("服务器链接失败！！！")
        print(err)
        # 如果报错使用sys的exit退出脚本
        sys.exit()
    # 使用exec_command方法执行命令，并使用变量接收命令的返回值并用print输出
    stdin, stdout, stderr = ssh_client.exec_command("df -hT")
    print(str(stdout.read()))

# 通过判断模块名运行上边函数
if __name__ == '__main__':
```

```
26        sshExeCMD()
```

运行结果：（错误的远程服务器主机地址）

```
/usr/local/python3/bin/python3.9 /home/test/PycharmProjects/hls_practice/27.py
服务器链接失败！！！
[Errno None] Unable to connect to port 22 on 192.168.1.110

Process finished with exit code 0
```

运行结果2：（错误的远程服务器主机登录密码）

```
/usr/local/python3/bin/python3.9 /home/test/PycharmProjects/hls_practice/27.py
服务器链接失败！！！
Authentication failed.

Process finished with exit code 0
```

多台远程服务器上执行命令

```python
# 导入paramiko，（导入前需要先在环境里安装该模块）
import paramiko
import sys

# 定义函数ssh,把操作内容写到函数里,函数里接收参数（写在括号里）,其中port=是设置一个默认值如果没传就用默认
def sshExeCMD(ip, username, password, port=22):
    # 定义一个变量ssh_clint使用SSHClient类用来后边调用
    ssh_client = paramiko.SSHClient()
    # 通过这个set_missing_host_key_policy方法用于实现登录是需要确认输入yes，否则保存
    ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    # 使用try做异常捕获
    try:
        # 使用connect类来连接服务器
        ssh_client.connect(hostname=ip, port=port, username=username, password=password)
    # 如果上边命令报错吧报错信息定义到err变量，并输出。
    except Exception as err:
        print("服务器链接失败！！！" % ip)
        print(err)
        # 如果报错使用sys的exit退出脚本
        sys.exit()
    # 使用exec_command方法执行命令，并使用变量接收命令的返回值并用print输出
    # 这里也可以把命令做成参数传进来
    stdin, stdout, stderr = ssh_client.exec_command("hostname")
    # 使用decode方法可以指定编码
    print(stdout.read().decode("utf-8"))

# 通过判断模块名运行上边函数
if __name__ == '__main__':
    # 定义一个字典，写服务器信息
    servers = {
        # 以服务器IP为键,值为服务器的用户密码端口定义的字典
```

```
        "192.168.1.110": {
            "username": "songxk",
            "password": "123123",
            "port": 22,
        },
        "192.168.1.123": {
            "username": "root",
            "password": "123123",
            "port": 22,
        },
    }
    # 使用items方法遍历，使用ip 和info把字典里的键和值遍历赋值，传给函数sshExeCMD
    for ip, info in servers.items():
        # 这里的info也就是上边定义的ip对应值的字典，使用get获取这个字典里对应username键对应的值，赋值给变量username传给函数中使用
        sshExeCMD(
            ip=ip,
            username=info.get("username"),
            password=info.get("password"),
            port=info.get("port")
        )
```

SSHClient 封装 Transport 连接远程服务器：

```
import paramiko

# 获取transport对象，配置主机名，端口
transport = paramiko.Transport(('172.16.1.1', 22))
# 设置登录名、密码
transport.connect(username='test', password='123')
# 获取ssh_client对象
ssh_client = paramiko.SSHClient()
ssh_client._transport = transport

# 获取远程服务器的主机名
stdin, stdout, stderr = ssh_client.exec_command('hostname')

res = stdout.read()
print(res.decode('utf-8'))

transport.close()
```

运行结果：

```
/usr/local/python3/bin/python3.9 /home/test/PycharmProjects/hls_practice/28.py
test-Vostro-3888-China-HDD-Protection


Process finished with exit code 0
```

基于公钥密钥连接远程服务器（本地主机通过私钥加密加密，远程服务器通过公钥解密）

1、客户端文件名：id_rsa（**id_rsa文件是本地主机的私钥，使用密钥连接远程服务器时，必须要在远程服务器上配制公钥**）

2、服务端必须有文件名：authorized_keys（**authorized_keys文件是远程服务器上的公钥**）（在用 `ssh-keygen` 时，必须制作一个authorized_keys，可以用 `ssh-copy-id` 来制作）

3、id_rsa文件的来源：终端执行命令 `ssh-keygen` ，然后一直回车，最后会在 `~/.ssh` 目录下生成该文件。

ssh客户端通过使用密钥连接远程服务器并执行命令获取控制台打印信息：

```python
import paramiko

private_key = paramiko.RSAKey.from_private_key_file('/tmp/id_rsa')

# 创建SSH对象
ssh = paramiko.SSHClient()

# 允许连接不在know_hosts文件中的主机
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

# 连接服务器
ssh.connect(hostname='120.92.84.249', port=22, username='root',
pkey=private_key)

# 执行命令
stdin, stdout, stderr = ssh.exec_command('df')

# 获取命令结果
result = stdout.read()
print(result.decode('utf-8'))

# 关闭连接
ssh.close()
```

SSHClient 封装 Transport 连接远程服务器：

```python
import paramiko

private_key = paramiko.RSAKey.from_private_key_file('/tmp/id_rsa')

# 获取transport对象，配置主机名，端口
transport = paramiko.Transport(('120.92.84.249', 22))

# 设置登录名、连接远程服务器的登录密钥
transport.connect(username='root', pkey=private_key)

# 获取ssh_client对象
ssh = paramiko.SSHClient()
ssh._transport = transport

```

```
# 获取远程服务器的主机名
stdin, stdout, stderr = ssh.exec_command('hostname')
result = stdout.read()
print(result.decode('utf-8'))

# 关闭连接
transport.close()
```

基于私钥字符串进行连接远程服务器：

```
import paramiko
from io import StringIO

key_str = """-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQEAsJmFLrSeCumJvga0Gl5O5wVOVwMIy2MpqIyQPi5J87dg89a4
Da9fczJog7qoSbRwHFOQoCHNphSlp5KPhGsF6RJewkIw9H1UKV4dCOyl/4HOAkAD
rKrsEDmrJ9JlzF2GTTZSnTgVQWcvBS2RKB4eM2R9aJ11xV6X2Hk4YDLTExIWeabb
h2TUKw0iyjI8pRuYLKkF2X16u9TBwfOTroGYgiNFHQvhsQppbEbI49NF2XkCkFMi
8/7tLjf95InE/VUUq56JqfzyHwdpHou+waXbwtvGgXN3sz+KkuEv6R2qDz06upZV
FCZRRpDhzoR8Uh/UEzTGZb8z7FB6EJXUiXJikQIBIwKCAQBBmBuGYFf1bK+BGG7H
9ySe81ecqVsJtx4aCFLVRGScWg4RbQKIvXs5an6XU/VdNGQnx0RYvBkvDvuzRRC8
J8Bd4kB0CfTtGJuaVigKoQp02HEWx1HSa17+tlWD0c4KFBvwywi+DYQ83S64x8gz
eOalX9bPFenqORPUD8R7gJeKvPVc6ZTPeorpuH7u9xayP0Eop8qKxZza9Xh3foVj
Qo4IxoYnDN57CIRX5PFSlDDggpmr8FtRF4nAxmFq8LhSp05ivzX/Ku1SNHdaMWZO
7va8tISXdLI5m0EGzoVoBvohIbwlxI6kfmamrh6Eas2Jnsc4CLzMsR4jBWt0LHLv
/SLnAoGBANaEUf/Jptab9G/xD9W2tw/636i3gLpTPY9KPtCcAxqStNeT6RAWZ5HF
lKJg+NKpu3pI45ldAwvts0i+aCZk2xakEWIZWqCmXm31JSPDQTaMGe7H0vOmUaxx
ncdpBVdvhMbfFUgei15iKfuafgrKaS9oIkntXEgrC+3wBOI0Gbx3AoGBANLAGxAF
TK7ydr+Q1+6/ujs6e8WsXt8HZMa/1khCVSbrf1MgACvZPSSSrDpVwaDTSjlRI4AL
bb0l0RFU+/0caMiHilscuJdz9Fdd9Ux4pjROZa3TF5CFhvP7PsZAoxOo+yqJg4zr
996GG/aAv4M8lQJ2rDFk/Dgn5y/AaAun1oM3AoGAGIQmoOPYjY4qkHNSRE9lYOl4
pZFQilKn8x5tlC8WTC4GCgJGhX7nQ9wQ/J1eQ/YkDfmznH+ok6YjHkGlgLsRuXHW
GdcDCwuzBUCWh76LHC1EytUCKnloa3qy8jfjWnMlHgrd3FtDILrC+C7p1Vj2FAvm
qVz0moiTpioPL8twp9MCgYEAin49q3EyZFYwxwdpU7/SJuvq750oZq0WVriUINsi
A6IR14oOvbqkhb94fhsY12ZGt/N9uosq22H+anms6CicoQicv4fnBHDFI3hCHE9I
pgeh50GTJHUA6Xk34V2s/kp5KpThazv6qCw+QubkQExh660SEdSlvoCfPKMCi1EJ
TukCgYAZKY1NZ2bjJyyO/dfNvMQ+etUL/9esi+40GUGyJ7SZcazrN9z+DO0yL39g
7FT9NMIc2dsmNJQMaGBCDl0AjO1O3b/wqlrNvNBGkanxn2Htn5ajfo+LBU7yHAcV
7w4X5HLarXiE1mj0LXFKJhdvFqU53KUQJXBqR6lsMqzsdPwLMJg==
-----END RSA PRIVATE KEY-----"""

private_key = paramiko.RSAKey(file_obj=StringIO(key_str))

# 获取transport对象，配置主机名，端口
transport = paramiko.Transport(('120.92.84.249', 22))

# 设置登录名、连接远程服务器的登录密钥
transport.connect(username='root', pkey=private_key)

# 获取ssh_client对象
ssh_client = paramiko.SSHClient()
ssh_client._transport = transport

```

```
# 远程在服务器上执行命令（获取远程服务器的主机名）
stdin, stdout, stderr = ssh_client.exec_command('df')

result = stdout.read()
print(result.decode('utf-8'))

# 关闭连接
transport.close()
```

paramiko 库中 SFTPClient 模块的使用

1、**用于连接远程服务器并执行上传下载文件**。

2、SFTPClient作为一个SFTP客户端对象，根据SSH传输协议的sftp会话，实现远程文件操作，比如文件上传、下载、权限、状态等操作。

3、常用方法：

`from_transport` 方法：# 创建一个已连通的SFTP客户端通道。

```
# 方法定义：
from_transport(cls,t)

# 参数说明：
t(transport),一个已通过验证的传输对象。
```

`get` 方法：从远程SFTP服务端下载文件到本地。

```
# 方法定义：
get(remotepath, localpath, callback=None)

# 参数说明：
remotepath(str类型)，需要下载的远程文件(源)；
callback(funcation(int,int)),获取已接收的字节数及总和传输字节数，以便回调函数调用，默认为None.
```

`SFTPClient` 类其它常用方法说明：

> mkdir：在SFTP服务端创建目录，如sftp.mkdir("/home/userdir",mode=0777),默认模式是0777(八进制)，在某些系统上，mode被忽略。在使用它的地方，当前的umask值首先被屏蔽掉。
>
> remove：删除SFTP服务端指定目录，如sftp.remove("/home/userdir")。
>
> rename：重命名SFTP服务端文件或目录，如sftp.rename("/home/test.sh","/home/testfile.sh")
>
> stat：获取远程SFTP服务端指定文件信息，如sftp.stat("/home/testfile.sh")。
>
> listdir：获取远程SFTP服务端指定目录列表，以Python的列表(List)形式返回，如sftp.listdir("/home")。

基于用户名密码的远程服务器文件的上传与下载

从远程服务器上上传或下载文件:

```python
import paramiko

# 与服务器创建ssh连接,transport方法建立通道，以元组的方式歇服务器信息
ssh_conn = paramiko.Transport(('120.92.84.249', 22))
ssh_conn.connect(username='root', password='xxx')

# 创建连接后，使用SFTPClient类和from_transport(括号里写上边创建的Transport通道)基于上边ssh连接创建一个sftp连接，定义成ftp_client变量后边方便引用
ftp_client = paramiko.SFTPClient.from_transport(ssh_conn)

# 下载文件
# ftp_client.get("远程服务器上的目标文件", "本地主机上的保存位置（需要具体到文件名）")
ftp_client.get("/etc/fstab", r"C:\Users\Administrator.USER-CH3G0KO3MG\Desktop\test\fstab")

# 上传文件
# ftp_client.put("本地主机上的文件位置", r"保存至远程服务器上的具体位置（需要具体到文件名）")
ftp_client.put(r"C:\Users\Administrator.USER-CH3G0KO3MG\Desktop\test\fstab", "/etc/fstab")

# 关闭ssh连接
ssh_conn.close()
```

基于公钥密钥的远程服务器的上传与下载

```python
import paramiko

private_key = paramiko.RSAKey.from_private_key_file('/tmp/id_rsa')

transport = paramiko.Transport(('120.92.84.249', 22))
transport.connect(username='root', pkey=private_key)

sftp = paramiko.SFTPClient.from_transport(transport)
# 将location.py 上传至服务器 /tmp/test.py
sftp.put('/tmp/id_rsa', '/tmp/a.txt')
# 将remove_path 下载到本地 local_path
sftp.get('remove_path', 'local_path')

transport.close()
```

实例1

```python
#!/usr/bin/env python
# -*- coding:utf-8 -*-
import paramiko
import uuid


def create_file():
    file_name = str(uuid.uuid4())
    with open(file_name, 'w') as f:
        f.write('sb')
    return file_name


class Haproxy(object):

    def __init__(self):
        self.host = '172.16.103.191'
        self.port = 22
        self.username = 'root'
        self.pwd = '123'
        self.__k = None

    def run(self):
        self.connect()
        self.upload()
        self.rename()
        self.close()

    def connect(self):
        transport = paramiko.Transport((self.host, self.port))
        transport.connect(username=self.username, password=self.pwd)
        self.__transport = transport

    def close(self):
        self.__transport.close()

    def upload(self):
        # 连接，上传
        file_name = create_file()

        sftp = paramiko.SFTPClient.from_transport(self.__transport)
        # 将location.py 上传至服务器 /tmp/test.py
        sftp.put(file_name, '/home/root/ttttttttttt.py')

    def rename(self):
        ssh = paramiko.SSHClient()
        ssh._transport = self.__transport
        # 执行命令
        stdin, stdout, stderr = ssh.exec_command('mv
/home/root/ttttttttttt.py /home/root/ooooooooo.py')
        # 获取命令结果
```

```
        result = stdout.read()
        return result.decode('utf-8')


ha = Haproxy()
ha.run()
```

实例2

```
def sshpc_copy_file(hostname, username, password, remote_path, local_path, put=False):
    """
    ssh到pc并拷贝文件到本机
    """
    try:
        t = paramiko.Transport(hostname, 22)
        # 连接远程服务器
        t.connect(username=username, password=password)
        # sftp传输协议
        sftp = paramiko.SFTPClient.from_transport(t)
        if put:
            sftp.put(local_path, remote_path)
        else:
            sftp.get(remote_path, local_path)
        t.close()
        return True
    except Exception as e:
        logger.debug(logger.log_info() + "SSH PC Error! Reason: " + str(e))
        return False
```

实例3

https://www.cnblogs.com/wztshine/p/11964321.html

实例4

> Python环境：3.9

目的

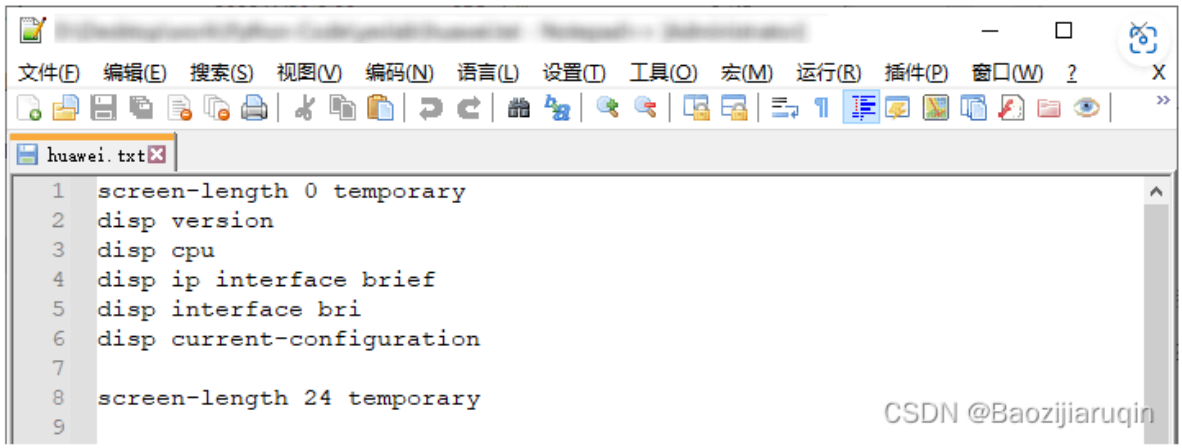
巡检设备，并将收集到的信息保存为文本。

拓扑

注：这里使用 EVE 模拟器，设备为 华为CE12800 交换机，通过桥接，电脑能正常远程到 华为CE12800

思路

1. 通过 `paramiko` 实例连接到交换机
2. 将巡检的命令写入到文本。



1. 读取文件后，遍历循环这些命令。
2. 收集回显信息，保存为文本。

交换机配置

```
system-view immediately
aaa
 undo local-user policy security-enhance
 local-user huawei password irreversible-cipher password
 local-user huawei service-type ssh
 local-user huawei level 3
quit

stelnet server enable
ssh user huawei
ssh user huawei authentication-type password
ssh user huawei service-type all
ssh authorization-type default aaa

user-interface vty 0 4
 authentication-mode aaa
quit

interface MEth0/0/0
 undo shutdown
 ip address 192.168.179.200 255.255.255.0
quit
```

完整代码

```
import paramiko
import time

ip = "192.168.179.200"
port = 22
username = "huawei"
password = "password"
```

```python

# 创建 SSHClient 实例
ssh_proc = paramiko.SSHClient()

# 自动添加主机名及主机密钥到本地HostKeys对象
# 如果不添加，那么不再本地know_hosts文件中记录的主机将无法连接
ssh_proc.set_missing_host_key_policy(paramiko.AutoAddPolicy())

# 连接交换机
ssh_proc.connect(hostname=ip, port=port, username=username,
password=password)

# 创建一个交互 shell 会话
shell = ssh_proc.invoke_shell()

# 读取 huawei.txt 文件，这里是一些需要指定的命令
with open('huawei.txt',mode='r') as f:
    # 读取文本内容，转换成列表，用于后面for 循环执行命令
    commands = list(f)
    # 列表推导式，删除文本中的空值
    commands = [x.strip('') for x in commands if x.strip() != '']

# 遍历循环列表
for i in commands:
    shell.send(i)
    time.sleep(0.5)

# 接受回显信息
data = shell.recv(999999).decode()

# 将接收的回显信息保存为“日期_logs.log”文件
with open(f'{time.strftime("%Y%m%d")}_logs.log',mode='w',encoding="utf-8")
as f:
    f.write(data)

print("收集完成！")
ssh_proc.close()
```

脚本执行完成后，当前路径下会生成一个 `20220421-logs.log` 文件。文件名会根据日期变动。

20220421-logs.log

```
1
2
3  Info: The max number of VTY users is 5, the number of current VTY users
4
5        The current login time is 2022-04-20 17:46:46.
6
7        The last login time is 2022-04-20 17:26:22 from 192.168.179.1 thr
8
9  <HUAWEI>screen-length 0 temporary
10
11 Info: The configuration takes effect on the current user terminal inter
12
13 <HUAWEI>disp version
14
15 Huawei Versatile Routing Platform Software
16
17 VRP (R) software, Version 8.180 (CE12800 V200R005C10SPC607B607)
18
19 Copyright (C) 2012-2018 Huawei Technologies Co., Ltd.
20
21 HUAWEI CE12800 uptime is 0 day, 1 hour, 46 minutes
22
23 SVRP Platform Version 1.0
24
25 <HUAWEI>disp cpu
26
27 CPU utilization statistics at 2022-04-20 17:46:47 980 ms
28
```

length : 6,071 lines : 363 Ln : 1 Col : 1 Pos : 1 Macintosh (CR) UTF-8 INS