使用脚本进行 ssh 登录服务器（Linux 新手之路）

前面的博客完成了一个简单的定时删除日志的脚本，但由于现在又多台服务器，需要在多台服务器上执行删除日志操作的脚本，有很多办法，我想到的方法， 一个是，可以将脚本部署到这些服务器上，然后在每台服务器上使用 crontab 来定时运行，但是如果服务器太多的话，就比较繁琐；还有一个办法就是通过脚 本去登录服务器，运行脚本，去完成操作，那怎样才能用脚本登录服务器呢？

这里主要要用到一个 Linux 的工具：expect，下面写一个简单的实例来测试，命名为 temp.sh

1. #!/usr/bin/expect
2. set timeout 10
3. spawn ssh -p 6022 auth@172.16.84.166
4. expect {
5. "*password:" {send "auth\r"}
6. }
7. interact

然后用 sh temp.sh,出现错误：spawn：command not found

通过查询资料，原来 expect 用的不是 bash，所以不能用 sh，通过./来执行：./temp.sh

出现错误：没有权限操作，那就改一下文件的权限：chmod 744 temp.sh

然后再：./temp.sh

然后成功登陆，这里先完成了一小步。

突然想起一个小问题，如果没有安装 expect 工具的，可以这样安装：

1. yum install expect

也有可能遇到问题，我遇到了如下问题：

没有找到公钥，不能安装

可以这样安装：

1. yum --nogpgcheck install expect

这样就可以跳过公钥，直接安装，结果图：

这样就安装成功了。（PS：需要清楚 yum 命令的，可以自己查一下）

通过以上的方式就可以实现通过脚本进行 SSH 登录服务器了。

简洁的一键 SSH 脚本

这里发一个自己图省事搞的一个批量打通 SSH 的脚本，可能对于好多朋友也是实用的,是 expect+python 的一个组合实现,原理非常 easy,
使用起来也不复杂,在此还是简单贴出来说说。

noscp.exp

```
#!/usr/bin/expect
#noscp.exp

if {$argc<4} {
 puts stderr "Usage: $argv0 localfile  remotefile user passwd "
 exit 1
}

set localfile [ lindex $argv 0 ]
set remotefile  [ lindex $argv 1 ]
set user  [ lindex $argv 2 ]
set pwd  [ lindex $argv 3 ]

set timeout 30
spawn scp ${localfile}  ${user}@${remotefile}

expect {
 "*yes/no" { send "yes\r"; exp_continue }
 "*password:" { send "$pwd\r" }
}
expect eof
```

ssh_distribute.py

```
#!/usr/bin/python

import subprocess
import os

file_dir='/home/hosts'

with open(file_dir) as data:
    for each_line in data.readlines():
        if each_line != '':
            (ip,passwd)=each_line.split(':',2)
            print('./noscp.exp ~/.ssh/authorized_keys '+ip+':~/.ssh '+'root '+passwd.strip('\n'))
            subprocess.Popen('./noscp.exp ~/.ssh/authorized_keys '+ip+':~/.ssh '+'root
'+passwd.strip('\n'),shell=True)
          #  subprocess.Popen('./sshkey.exp '+ip+' root '+passwd+' \\| grep ssh-rsa >>
~/.ssh/authorized_keys',shell=True)
        else:
            pass
#subprocess.Popen('chmod 755 ~/.ssh/authorized_keys',shell=True)
```

ssh_setup.py

```
#!/usr/bin/python

import subprocess
import os

file_dir='/home/hosts'

with open(file_dir) as data:
    for each_line in data.readlines():
        if each_line != '':
            (ip,passwd)=each_line.split(':',2)
            print('./sshkey.exp '+ip+' root '+passwd.strip('\n')+' | grep ssh-rsa >> ~/.ssh/authorized_keys')
            subprocess.Popen('./sshkey.exp '+ip+' root '+passwd.strip('\n')+' | grep ssh-rsa >>
~/.ssh/authorized_keys',shell=True)
          #  subprocess.Popen('./sshkey.exp '+ip+' root '+passwd+' \\| grep ssh-rsa >>
~/.ssh/authorized_keys',shell=True)
        else:
            pass
subprocess.Popen('chmod 755 ~/.ssh/authorized_keys',shell=True)
#subprocess.Popen('/home/ssh_distribute.py',shell=True)
```

sshkey.exp

```
#!/usr/bin/expect

#sshkey.exp
```

```
if {$argc<3} {
 puts stderr "Usage: $argv0 host  user  passwd "
 exit 1
}

set host [ lindex $argv 0 ]
set user  [ lindex $argv 1 ]
set pwd  [ lindex $argv 2 ]

set timeout 30



#spawn ssh  ${user}@${host} "rm -rf ~/.ssh/id_rsa*"
#
#expect {
# "*yes/no" { send "yes\r"; exp_continue }
# "*password:" { send "$pwd\r"; exp_continue  }
#}


spawn ssh  ${user}@${host} "ssh-keygen -t rsa"

expect {
 "*yes/no" { send "yes\r"; exp_continue }
 "*password:" { send "$pwd\r"; exp_continue  }
 "Enter file in which to save the key*" { send "\n\r"; exp_continue }
 "Overwrite*" { send "y\n"; exp_continue }
 "Enter passphrase (empty for no passphrase):" { send "\n\r"; exp_continue }
 "Enter same passphrase again:" { send "\n\r" }
}

spawn ssh  ${user}@${host} "cat ~/.ssh/id_rsa.pub"

expect {
 "*yes/no" { send "yes\r"; exp_continue }
 "*password:" { send "$pwd\r"  }
}

expect eof
```

多看两眼代码应该能够看出,expect 的功能是能够等待一些 Linux 反馈 通过这个的反馈做出推断并能够分类进行兴许的动作，非常黄非常暴力。

也就是利用了这个原理。过程例如以下:

1.首先运行 ./ssh_setup.py 首先收集全部机器的公钥,然后定向到运行这个脚本的 authorized_keys 文件中边,自己主动赋予 755 权限。

2.运行./ssh_distribute.py 分发 authorized_keys 文件到全部的机器上。

下载连接在下方,详细用法里边有 readme.txt

http://download.csdn.net/detail/u012886375/9453810