Linux 远程批量工具 mooon_ssh 和 mooon_upload 使用示例

一见 2018/9/9

目录

1. 前言

远程批量工具包含：

1) 批量命令工具 mooon_ssh；
2) 批量上传文件工具 mooon_upload；
3) 批量下载文件工具 mooon_download。

可执行二进制包下载地址：
https://github.com/eyjian/libmooon/releases

源代码包下载地址：
https://github.com/eyjian/libmooon/archive/master.zip

批量工具除由三个工具组成外，还分两个版本：

1) C++版本
2) GO 版本

当前 C++版本比较成熟，GO 版本相当简略，但 C++版本依赖 C++运行时库，不同环境

需要特定编译，而 GO 版本可不依赖 C 和 C++运行时库，所以不需编译即可应用到广泛的 Linux 环境。

使用简单，直接执行命令，即会提示用法，如 C++版本：

```
$ mooon_ssh
parameter[-c]'s value not set

usage:
-h[]: Connect to the remote machines on the given hosts separated by comma, can be replaced by environment variable 'H', example: -h='192.168.1.10,192.168.1.11'
-P[36000/10,65535]: Specifies the port to connect to on the remote machines, can be replaced by environment variable 'PORT'
-u[]: Specifies the user to log in as on the remote machines, can be replaced by environment variable 'U'
-p[]: The password to use when connecting to the remote machines, can be replaced by environment variable 'P'
-t[60/1,65535]: The number of seconds before connection timeout
-c[]: The command is executed on the remote machines, example: -c='grep ERROR /tmp/*.log'
-v[1/0,2]: Verbosity, how much troubleshooting info to print
```

2. 批量执行命令工具：mooon_ssh

参数名	默认值	说明
-u	无	用户名参数，可用环境变量 U 替代
-p	无	密码参数，可用环境变量 P 替代
-h	无	IP 列表参数，可用环境变量 H 替代
-P	22，可修改源码，编译为常用端口号	SSH 端口参数，可用环境变量 PORT 替代
-c	无	在远程机器上执行的命令，建议单引号方式指定值，除非要执行的命令本身已经包含了单引号有冲突。使用双引号时，要注意转义，否则会被本地 shell 解释
-v	1	工具输出的详细度

3. 批量上传文件工具：mooon_upload

参数名	默认值	说明
-u	无	用户名参数，可用环境变量 U 替代
-p	无	密码参数，可用环境变量 P 替代
-h	无	IP 列表参数，可用环境变量 H 替代
-P	22，可修改源	SSH 端口参数，可用环境变量 PORT 替代

	码，编译为常用端口号	
-s	无	以逗号分隔的，需要上传的本地文件列表，可以带相对或绝对目录
-d	无	文件上传到远程机器的目录，只能为单个目录

4. 使用示例

4.1. 使用示例 1：上传/etc/hosts

```
mooon_upload -s=/etc/hosts -d=/etc
```

4.2. 使用示例 2：检查/etc/profile 文件是否一致

```
mooon_ssh -c='md5sum /etc/hosts'
```

4.3. 使用示例 3：批量查看 crontab

```
mooon_ssh -c='crontab -l'
```

4.4. 批量清空 crontab

```
mooon_ssh -c='rm -f /tmp/crontab.empty;touch /tmp/crontab.empty'
mooon_ssh -c='crontab /tmp/crontab.emtpy'
```

4.5. 批量更新 crontab

```
mooon_ssh -c='crontab /tmp/crontab.online'
```

4.6. 使用示例 4：取远端机器 IP

因为 awk 用单引号，所以参数“-c”的值不能使用单引号，所以内容需要转义，相对其它来说要复杂点：

```
mooon_ssh -c="netstat -ie | awk -F[\\ :]+ 'BEGIN{ok=0;}{if (match(\$0, \"eth1\")) ok=1; if ((1==ok) && match(\$0,\"inet\")) { ok=0; if (7==NF) printf(\"%s\\n\",\$3); else printf(\"%s\\n\",\$4);} }'"
```

不同的环境，IP 在“netstat -ie”输出中的位置稍有不同，所以 awk 中加了“7==NF”判断，但仍不一定适用于所有的环境。需要转义的字符包含：双引号、美元符和斜杠。

4.7. 使用示例 5：批量查看 kafka 进程（环境变量方式）

```
$ export H=192.168.31.9,192.168.31.10,192.168.31.11,192.168.31.12,192.168.31.13
$ export U=kafka
$ export P='123456'

$ mooon_ssh -c='/usr/local/jdk/bin/jps -m'
[192.168.31.15]
50928 Kafka /data/kafka/config/server.properties
125735 Jps -m
[192.168.31.15] SUCCESS

[192.168.31.16]
147842 Jps -m
174902 Kafka /data/kafka/config/server.properties
[192.168.31.16] SUCCESS

[192.168.31.17]
51409 Kafka /data/kafka/config/server.properties
178771 Jps -m
[192.168.31.17] SUCCESS

[192.168.31.18]
73568 Jps -m
62314 Kafka /data/kafka/config/server.properties
[192.168.31.18] SUCCESS

[192.168.31.19]
123908 Jps -m
182845 Kafka /data/kafka/config/server.properties
[192.168.31.19] SUCCESS


================================
[192.168.31.15 SUCCESS] 0 seconds
[192.168.31.16 SUCCESS] 0 seconds
[192.168.31.17 SUCCESS] 0 seconds
[192.168.31.18 SUCCESS] 0 seconds
[192.168.31.19 SUCCESS] 0 seconds
SUCCESS: 5, FAILURE: 0
```

4.8. 使用示例 6：批量停止 kafka 进程（参数方式）

```
$ mooon_ssh -c='/data/kafka/bin/kafka-server-stop.sh' -u=kafka -p='123456'
-h=192.168.31.15,192.168.31.16,192.168.31.17,192.168.31.18,192.168.31.19
[192.168.31.15]
No kafka server to stop
command return 1

[192.168.31.16]
No kafka server to stop
command return 1

[192.168.31.17]
No kafka server to stop
command return 1

[192.168.31.18]
No kafka server to stop
command return 1

[192.168.31.19]
No kafka server to stop
command return 1

================================
[192.168.31.15 FAILURE] 0 seconds
[192.168.31.16 FAILURE] 0 seconds
[192.168.31.17 FAILURE] 0 seconds
[192.168.31.18 FAILURE] 0 seconds
[192.168.31.19 FAILURE] 0 seconds
SUCCESS: 0, FAILURE: 5
```

5. 如何编译批量工具？

5.1. GO 版本

依赖的 crypto 包，从 https://github.com/golang/crypto 下载，放到目录 $GOPATH/src/golang.org/x 或 $GOROOT/src/golang.org/x 下。注意需要先创建好目录 $GOROOT/src/golang.org/x，然后在此目录下解压 crypto 包。如果下载的包名为 crypto-master.zip，则解压后的目录名为 crypto-master，需要重命名为 crypto。

安装 crypto 包示例：

```
1）安装 go
cd /usr/local
tar xzf go1.10.3.linux-386.tar.gz
2）mkdir -p go/golang.org/x
3）cd go/golang.org/x
4）unzip crypto-master.zip
5）mv crypto-master crypto
```

命令行执行“go help gopath”可了解 gopath，或执行“go env”查看当前的设置。编译方法：

```
go build -o mooon_ssh mooon_ssh.go
```

上述编译会依赖 glibc，如果不想依赖，这样编译：

```
go build -o mooon_ssh -ldflags '-linkmode "external" -extldflags "-static"' mooon_ssh.go
```

5.2. C++版本

C++版本为 libmooon 组成部分，编译 libmooon 即可得到 mooon_ssh、mooon_upload 和 mooon_download。但 libmooon 依赖 libssh2，而 libssh2 又依赖 openssl，所以需要先依次安装好 openssl 和 libssh2。

libssh2 下载地址：http://www.libssh2.org。

1) openssl 编译安装方法

解压后进入 openssl 源码目录，以版本 openssl-1.0.2i 为例，依次执行：

```
./config --prefix=/usr/local/openssl-1.0.2i shared threads
make
make install
ln -s /usr/local/openssl-1.0.2i /usr/local/openssl
```

2) libssh2 编译安装方法

解压后进入 libssh2 源码目录，以版本 libssh2-1.6.0 为例，依次执行：

```
./configure --prefix=/usr/local/libssh2-1.6.0 --with-libssl-prefix=/usr/local/openssl
make
make install
```

注意：libssh2 和比较新版本的 openssl 可能存在兼容问题。

3) libmooon 编译方法

采用 cmake 编译，所以需要先安装好 cmake，并要求 cmake 版本不低于 2.8.11，可执行“cmake --version”查看 cmake 版本，当 cmake、libssh2 和 openssl 准备好后执行下列命令编译 libmooon 即可得到批量工具：

```
cmake -DCMAKE_BUILD_TYPE=Debug -DCMAKE_INSTALL_PREFIX=/usr/local/mooon .
make
```

```
make install
```

在 make 一步成功后，即可在 tools 子目录中找到 mooon_ssh、mooon_upload 和 mooon_download，实践中 mooon_download 可能使用得比较少。