curl使用

在Linux中curl是一个利用URL规则在命令行下工作的文件传输工具，可以说是一款很强大的http命令行工具。它支持文件的上传和下载，是综合传输工具，但按传统，习惯称url为下载工具。

```
-A/--user-agent <string>              设置用户代理发送给服务器
-b/--cookie <name=string/file>     cookie字符串或文件读取位置
-c/--cookie-jar <file>                操作结束后把cookie写入到这个文件中
-C/--continue-at <offset>             断点续转
-D/--dump-header <file>               把header信息写入到该文件中
-e/--referer                          来源网址
-f/--fail                             连接失败时不显示http错误
-o/--output                           把输出写到该文件中
-O/--remote-name                      把输出写到该文件中，保留远程文件的文件名
-r/--range <range>                    检索来自HTTP/1.1或FTP服务器字节范围
-s/--silent                           静音模式。不输出任何东西
-T/--upload-file <file>               上传文件
-u/--user <user[:password]>       设置服务器的用户和密码
-w/--write-out [format]               什么输出完成后
-x/--proxy <host[:port]>              在给定的端口上使用HTTP代理
-#/--progress-bar                     进度条显示当前的传送状态
```

例子：

1、基本用法

```
# curl http://www.linux.com
```

执行后，www.linux.com 的html就会显示在屏幕上了
Ps：由于安装linux的时候很多时候是没有安装桌面的，也意味着没有浏览器，因此这个方法也经常用于测试一台服务器是否可以到达一个网站

2、保存访问的网页
2.1:使用linux的重定向功能保存

```
# curl http://www.linux.com >> linux.html
```

2.2:可以使用curl的内置option:-o(小写)保存网页

```
curl -o linux.html http://www.linux.com
```

执行完成后会显示如下界面，显示100%则表示保存成功

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 79684    0 79684    0     0  3437k      0 --:--:-- --:--:-- --:--:-- 7781k
```

2.3:可以使用curl的内置option:-O(大写)保存网页中的文件
要注意这里后面的url要具体到某个文件，不然抓不下来

```
# curl -O http://www.linux.com/hello.sh
```

3、测试网页返回值

```
# curl -o /dev/null -s -w %{http_code} www.linux.com
```

Ps:在脚本中，这是很常见的测试网站是否正常的用法

4、指定proxy服务器以及其端口
很多时候上网需要用到代理服务器(比如是使用代理服务器上网或者因为使用curl别人网站而被别人屏蔽IP地址的时候)，幸运的是curl通过使用内置option：-x来支持设置代理

```
# curl -x 192.168.100.100:1080 http://www.linux.com
```

5、cookie
有些网站是使用cookie来记录session信息。对于chrome这样的浏览器，可以轻易处理cookie信息，但在curl中只要增加相关参数也是可以很容易的处理cookie
5.1:保存http的response里面的cookie信息。内置option:-c（小写）

```
# curl -c cookiec.txt  http://www.linux.com
```

执行后cookie信息就被存到了cookiec.txt里面了

5.2:保存http的response里面的header信息。内置option: -D

```
# curl -D cookied.txt http://www.linux.com
```

执行后cookie信息就被存到了cookied.txt里面了

注意：-c(小写)产生的cookie和-D里面的cookie是不一样的。

5.3:使用cookie
很多网站都是通过监视你的cookie信息来判断你是否按规矩访问他们的网站的，因此我们需要使用保存的cookie信息。内置option: -b

```
# curl -b cookiec.txt http://www.linux.com
```

6、模仿浏览器
有些网站需要使用特定的浏览器去访问他们，有些还需要使用某些特定的版本。curl内置option:-A可以让我们指定浏览器去访问网站

```
# curl -A "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.0)"
http://www.linux.com
```

这样服务器端就会认为是使用IE8.0去访问的

7、伪造referer（盗链）
很多服务器会检查http访问的referer从而来控制访问。比如：你是先访问首页，然后再访问首页中的邮箱页面，这里访问邮箱的referer地址就是访问首页成功后的页面地址，如果服务器发现对邮箱页面访问的referer地址不是首页的地址，就断定那是个盗连了
curl中内置option：-e可以让我们设定referer

```
# curl -e "www.linux.com" http://mail.linux.com
```

这样就会让服务器其以为你是从www.linux.com点击某个链接过来的

8、下载文件
8.1：利用curl下载文件。
#使用内置option：-o(小写)

```
# curl -o dodo1.jpg http:www.linux.com/dodo1.JPG
```

#使用内置option：-O（大写)

```
# curl -O http://www.linux.com/dodo1.JPG
```

这样就会以服务器上的名称保存文件到本地

8.2：循环下载
有时候下载图片可以能是前面的部分名称是一样的，就最后的尾椎名不一样

```
# curl -O http://www.linux.com/dodo[1-5].JPG
```

这样就会把dodo1，dodo2，dodo3，dodo4，dodo5全部保存下来

8.3：下载重命名

```
# curl -O http://www.linux.com/{hello,bb}/dodo[1-5].JPG
```

由于下载的hello与bb中的文件名都是dodo1，dodo2，dodo3，dodo4，dodo5。因此第二次下载的会把第一次下载的覆盖，这样就需要对文件进行重命名。

```
# curl -o #1_#2.JPG http://www.linux.com/{hello,bb}/dodo[1-5].JPG
```

这样在hello/dodo1.JPG的文件下载下来就会变成hello_dodo1.JPG,其他文件依此类推，从而有效的避免了文件被覆盖

8.4：分块下载
有时候下载的东西会比较大，这个时候我们可以分段下载。使用内置option：-r

```
# curl -r 0-100 -o dodo1_part1.JPG http://www.linux.com/dodo1.JPG
# curl -r 100-200 -o dodo1_part2.JPG http://www.linux.com/dodo1.JPG
# curl -r 200- -o dodo1_part3.JPG http://www.linux.com/dodo1.JPG
# cat dodo1_part* > dodo1.JPG
```

这样就可以查看dodo1.JPG的内容了

8.5：通过ftp下载文件
curl可以通过ftp下载文件，curl提供两种从ftp中下载的语法

```
# curl -O -u 用户名:密码 ftp://www.linux.com/dodo1.JPG
# curl -O ftp://用户名:密码@www.linux.com/dodo1.JPG
```

8.6：显示下载进度条

```
# curl -# -O http://www.linux.com/dodo1.JPG
```

8.7：不会显示下载进度信息

```
# curl -s -O http://www.linux.com/dodo1.JPG
```

9、断点续传
在windows中，我们可以使用迅雷这样的软件进行断点续传。curl可以通过内置option:-C同样可以达到相同的效果
如果在下载dodo1.JPG的过程中突然掉线了，可以使用以下的方式续传

```
# curl -C -O http://www.linux.com/dodo1.JPG
```

10、上传文件
curl不仅仅可以下载文件，还可以上传文件。通过内置option:-T来实现

```
# curl -T dodo1.JPG -u 用户名:密码 ftp://www.linux.com/img/
```

这样就向ftp服务器上传了文件dodo1.JPG

11、显示抓取错误

```
# curl -f http://www.linux.com/error
```

其他参数：

```
-a/--append                        上传文件时，附加到目标文件
--anyauth                          可以使用“任何”身份验证方法
--basic                            使用HTTP基本验证
-B/--use-ascii                     使用ASCII文本传输
-d/--data <data>                   HTTP POST方式传送数据
--data-ascii <data>                以ascii的方式post数据
--data-binary <data>               以二进制的方式post数据
--negotiate                        使用HTTP身份验证
--digest                           使用数字身份验证
--disable-eprt                     禁止使用EPRT或LPRT
--disable-epsv                     禁止使用EPSV
--egd-file <file>                  为随机数据(SSL)设置EGD socket路径
--tcp-nodelay                      使用TCP_NODELAY选项
-E/--cert <cert[:passwd]>          客户端证书文件和密码 (SSL)
--cert-type <type>                 证书文件类型 (DER/PEM/ENG) (SSL)
--key <key>                        私钥文件名 (SSL)
--key-type <type>                  私钥文件类型 (DER/PEM/ENG) (SSL)
--pass  <pass>                     私钥密码 (SSL)
--engine <eng>                     加密引擎使用 (SSL). "--engine list" for list
--cacert <file>                    CA证书 (SSL)
--capath <directory>               CA目  (made using c_rehash) to verify peer
against (SSL)
--ciphers <list>                   SSL密码
--compressed                       要求返回是压缩的形势 (using deflate or gzip)
--connect-timeout <seconds>        设置最大请求时间
--create-dirs                      建立本地目录的目录层次结构
--crlf                             上传是把LF转变成CRLF
--ftp-create-dirs                  如果远程目录不存在，创建远程目录
```

```
--ftp-method [multicwd/nocwd/singlecwd]     控制CWD的使用
--ftp-pasv                                  使用 PASV/EPSV 代替端口
--ftp-skip-pasv-ip                          使用PASV的时候,忽略该IP地址
--ftp-ssl                                   尝试用 SSL/TLS 来进行ftp数据传输
--ftp-ssl-reqd                              要求用 SSL/TLS 来进行ftp数据传输
-F/--form <name=content>                    模拟http表单提交数据
-form-string <name=string>                  模拟http表单提交数据
-g/--globoff                                禁用网址序列和范围使用{}和[]
-G/--get                                    以get的方式来发送数据
-h/--help                                   帮助
-H/--header <line>                          自定义头信息传递给服务器
--ignore-content-length                     忽略的HTTP头信息的长度
-i/--include                                输出时包括protocol头信息
-I/--head                                   只显示文档信息
-j/--junk-session-cookies                   读取文件时忽略session cookie
--interface <interface>                     使用指定网络接口/地址
--krb4 <level>                              使用指定安全级别的krb4
-k/--insecure                               允许不使用证书到SSL站点
-K/--config                                 指定的配置文件读取
-l/--list-only                              列出ftp目录下的文件名称
--limit-rate <rate>                         设置传输速度
--local-port<NUM>                           强制使用本地端口号
-m/--max-time <seconds>                     设置最大传输时间
--max-redirs <num>                          设置最大读取的目录数
--max-filesize <bytes>                      设置最大下载的文件总量
-M/--manual                                 显示全手动
-n/--netrc                                  从netrc文件中读取用户名和密码
--netrc-optional                            使用 .netrc 或者 URL来覆盖-n
--ntlm                                      使用 HTTP NTLM 身份验证
-N/--no-buffer                              禁用缓冲输出
-p/--proxytunnel                            使用HTTP代理
--proxy-anyauth                             选择任一代理身份验证方法
--proxy-basic                               在代理上使用基本身份验证
--proxy-digest                              在代理上使用数字身份验证
--proxy-ntlm                                在代理上使用ntlm身份验证
-P/--ftp-port <address>                     使用端口地址，而不是使用PASV
-Q/--quote <cmd>                            文件传输前，发送命令到服务器
--range-file                                读取（SSL）的随机文件
-R/--remote-time                            在本地生成文件时，保留远程文件时间
--retry <num>                               传输出现问题时，重试的次数
--retry-delay <seconds>                     传输出现问题时，设置重试间隔时间
--retry-max-time <seconds>                  传输出现问题时，设置最大重试时间
-S/--show-error                             显示错误
--socks4 <host[:port]>                      用socks4代理给定主机和端口
--socks5 <host[:port]>                      用socks5代理给定主机和端口
-t/--telnet-option <OPT=val>                Telnet选项设置
--trace <file>                              对指定文件进行debug
--trace-ascii <file>                        Like --跟踪但没有hex输出
--trace-time                                跟踪/详细输出时，添加时间戳
--url <URL>                                 Spet URL to work with
-U/--proxy-user <user[:password]>           设置代理用户名和密码
-V/--version                                显示版本信息
-X/--request <command>                      指定什么命令
-y/--speed-time                             放弃限速所要的时间。默认为30
-Y/--speed-limit                            停止传输速度的限制，速度时间'秒
-z/--time-cond                              传送时间设置
-0/--http1.0                                使用HTTP 1.0
-1/--tlsv1                                  使用TLSv1（SSL）
```

```
-2/--sslv2                      使用SSLv2的（SSL）
-3/--sslv3                      使用的SSLv3（SSL）
--3p-quote                      like -Q for the source URL for 3rd party
transfer
--3p-url                        使用url，进行第三方传送
--3p-user                       使用用户名和密码，进行第三方传送
-4/--ipv4                       使用IP4
-6/--ipv6                       使用IP6
```