curl

利用URL规则在命令行下工作的文件传输工具

补充说明

curl命令 是一个利用URL规则在命令行下工作的文件传输工具。它支持文件的上传和下载，所以是综合传输工具，但按传统，习惯称curl为下载工具。作为一款强力工具，curl支持包括HTTP、HTTPS、

语法

```
curl(选项)(参数)
```

选项

```
<table border="0" cellpadding="0" cellspacing="0">

<tbody>
<tr><td>-a/--append</td><td>上传文件时，附加到目标文件</td></tr>
<tr><td>-A/--user-agent <string></td><td>设置用户代理发送给服务器</td></tr>
<tr><td>-anyauth</td><td>可以使用“任何”身份验证方法</td></tr>
<tr><td>-b/--cookie <name=string/file></td><td>cookie字符串或文件读取位
置</td></tr>
<tr><td>     --basic</td><td>使用HTTP基本验证</td></tr>
<tr><td>-B/--use-ascii</td><td>使用ASCII /文本传输</td></tr>
<tr><td>-c/--cookie-jar <file></td><td>操作结束后把cookie写入到这个文件
中</td></tr>
<tr><td>-C/--continue-at <offset></td><td>断点续传</td></tr>
<tr><td>-d/--data <data></td><td>HTTP POST方式传送数据</td></tr>
<tr><td>     --data-ascii <data></td><td>以ascii的方式post数据</td></tr>
<tr><td>     --data-binary <data></td><td>以二进制的方式post数据</td></tr>
<tr><td>     --negotiate</td><td>使用HTTP身份验证</td></tr>
<tr><td>     --digest</td><td>使用数字身份验证</td></tr>
<tr><td>     --disable-eprt</td><td>禁止使用EPRT或LPRT</td></tr>
<tr><td>     --disable-epsv</td><td>禁止使用EPSV</td></tr>
<tr><td>-D/--dump-header <file></td><td>把header信息写入到该文件中</td></tr>
<tr><td>     --egd-file <file></td><td>为随机数据(SSL)设置EGD socket路
径</td></tr>
<tr><td>     --tcp-nodelay</td><td>使用TCP_NODELAY选项</td></tr>
<tr><td>-e/--referer</td><td>来源网址</td></tr>
<tr><td>-E/--cert <cert:[passwd]></td><td>客户端证书文件和密码 (SSL)</td></tr>
<tr><td>     --cert-type <type></td><td>证书文件类型 (DER/PEM/ENG)
(SSL)</td></tr>
<tr><td>     --key <key></td><td>私钥文件名 (SSL)</td></tr>
<tr><td>     --key-type <type></td><td>私钥文件类型 (DER/PEM/ENG)
(SSL)</td></tr>
<tr><td>     --pass <pass></td><td>私钥密码 (SSL)</td></tr>
<tr><td>     --engine <eng></td><td>加密引擎使用 (SSL). "--engine list" for
list</td></tr>
<tr><td>     --cacert <file></td><td>CA证书 (SSL)</td></tr>
```

```
<tr><td>     --capath <directory></td><td>CA目录 (made using c_rehash) to
verify peer against (SSL)</td></tr>
<tr><td>     --ciphers <list></td><td>SSL密码</td></tr>
<tr><td>     --compressed</td><td>要求返回是压缩的形势 (using deflate or
gzip)</td></tr>
<tr><td>     --connect-timeout <seconds></td><td>设置最大请求时间</td></tr>
<tr><td>     --create-dirs</td><td>建立本地目录的目录层次结构</td></tr>
<tr><td>     --crlf</td><td>上传是把LF转变成CRLF</td></tr>
<tr><td>-f/--fail</td><td>连接失败时不显示http错误</td></tr>
<tr><td>     --ftp-create-dirs</td><td>如果远程目录不存在，创建远程目录</td></tr>
<tr><td>     --ftp-method [multicwd/nocwd/singlecwd]</td><td>控制CWD的使
用</td></tr>
<tr><td>     --ftp-pasv</td><td>使用 PASV/EPSV 代替端口</td></tr>
<tr><td>     --ftp-skip-pasv-ip</td><td>使用PASV的时候,忽略该IP地址</td></tr>
<tr><td>     --ftp-ssl</td><td>尝试用 SSL/TLS 来进行ftp数据传输</td></tr>
<tr><td>     --ftp-ssl-reqd</td><td>要求用 SSL/TLS 来进行ftp数据传输</td></tr>
<tr><td>-F/--form <name=content></td><td>模拟http表单提交数据</td></tr>
<tr><td>     --form-string <name=string></td><td>模拟http表单提交数
据</td></tr>
<tr><td>-g/--globoff</td><td>禁用网址序列和范围使用{}和[]</td></tr>
<tr><td>-G/--get</td><td>以get的方式来发送数据</td></tr>
<tr><td>-H/--header <line></td><td>自定义头信息传递给服务器</td></tr>
<tr><td>     --ignore-content-length</td><td>忽略的HTTP头信息的长度</td></tr>
<tr><td>-i/--include</td><td>输出时包括protocol头信息</td></tr>
<tr><td>-I/--head</td><td>只显示请求头信息</td></tr>
<tr><td>-j/--junk-session-cookies</td><td>读取文件进忽略session
cookie</td></tr>
<tr><td>     --interface <interface></td><td>使用指定网络接口/地址</td></tr>
<tr><td>     --krb4 <level></td><td>使用指定安全级别的krb4</td></tr>
<tr><td>-k/--insecure</td><td>允许不使用证书到SSL站点</td></tr>
<tr><td>-K/--config</td><td>指定的配置文件读取</td></tr>
<tr><td>-l/--list-only</td><td>列出ftp目录下的文件名称</td></tr>
<tr><td>     --limit-rate <rate></td><td>设置传输速度</td></tr>
<tr><td>     --local-port<NUM></td><td>强制使用本地端口号</td></tr>
<tr><td>-m/--max-time <seconds></td><td>设置最大传输时间</td></tr>
<tr><td>     --max-redirs <num></td><td>设置最大读取的目录数</td></tr>
<tr><td>     --max-filesize <bytes></td><td>设置最大下载的文件总量</td></tr>
<tr><td>-M/--manual</td><td>显示全手动</td></tr>
<tr><td>-n/--netrc</td><td>从netrc文件中读取用户名和密码</td></tr>
<tr><td>     --netrc-optional</td><td>使用 .netrc 或者 URL来覆盖-n</td></tr>
<tr><td>     --ntlm</td><td>使用 HTTP NTLM 身份验证</td></tr>
<tr><td>-N/--no-buffer</td><td>禁用缓冲输出</td></tr>
<tr><td>-o/--output</td><td>把输出写到该文件中</td></tr>
<tr><td>-O/--remote-name</td><td>把输出写到该文件中，保留远程文件的文件
名</td></tr>
<tr><td>-p/--proxytunnel</td><td>使用HTTP代理</td></tr>
<tr><td>     --proxy-anyauth</td><td>选择任一代理身份验证方法</td></tr>
<tr><td>     --proxy-basic</td><td>在代理上使用基本身份验证</td></tr>
<tr><td>     --proxy-digest</td><td>在代理上使用数字身份验证</td></tr>
<tr><td>     --proxy-ntlm</td><td>在代理上使用ntlm身份验证</td></tr>
<tr><td>-P/--ftp-port <address></td><td>使用端口地址，而不是使用PASV</td></tr>
```

```
<tr><td>-q</td><td>作为第一个参数，关闭 .curlrc</td></tr>
<tr><td>-Q/--quote <cmd></td><td>文件传输前，发送命令到服务器</td></tr>
<tr><td>-r/--range <range></td><td>检索来自HTTP/1.1或FTP服务器字节范围</td></tr>
<tr><td>--range-file</td><td>读取□SSL□的随机文件</td></tr>
<tr><td>-R/--remote-time</td><td>在本地生成文件时，保留远程文件时间</td></tr>
<tr><td>     --retry <num></td><td>传输出现问题时，重试的次数</td></tr>
<tr><td>     --retry-delay <seconds></td><td>传输出现问题时，设置重试间隔时
间</td></tr>
<tr><td>     --retry-max-time <seconds></td><td>传输出现问题时，设置最大重试时
间</td></tr>
<tr><td>-s/--silent</td><td>静默模式。不输出任何东西</td></tr>
<tr><td>-S/--show-error</td><td>显示错误</td></tr>
<tr><td>     --socks4 <host[:port]></td><td>用socks4代理给定主机和端
口</td></tr>
<tr><td>     --socks5 <host[:port]></td><td>用socks5代理给定主机和端
口</td></tr>
<tr><td>     --stderr <file></td><td> </td></tr>
<tr><td>-t/--telnet-option <OPT=val></td><td>Telnet选项设置</td></tr>
<tr><td>     --trace <file></td><td>对指定文件进行debug</td></tr>
<tr><td>     --trace-ascii <file></td><td>Like --跟踪但没有hex输出</td></tr>
<tr><td>     --trace-time</td><td>跟踪/详细输出时，添加时间戳</td></tr>
<tr><td>-T/--upload-file <file></td><td>上传文件</td></tr>
<tr><td>     --url <URL></td><td>Spet URL to work with</td></tr>
<tr><td>-u/--user <user[:password]></td><td>设置服务器的用户和密码</td></tr>
<tr><td>-U/--proxy-user <user[:password]></td><td>设置代理用户名和密
码</td></tr>
<tr><td>-w/--write-out [format]</td><td>什么输出完成后</td></tr>
<tr><td>-x/--proxy <host[:port]></td><td>在给定的端口上使用HTTP代理</td></tr>
<tr><td>-X/--request <command></td><td>指定什么命令</td></tr>
<tr><td>-y/--speed-time</td><td>放弃限速所要的时间，默认为30</td></tr>
<tr><td>-Y/--speed-limit</td><td>停止传输速度的限制，速度时间</td></tr>
</tbody>

</table>
```

实例

文件下载

curl命令可以用来执行下载、发送各种HTTP请求，指定HTTP头部等操作。如果系统没有curl可以使用`yum install curl`安装，也可以下载安装□curl是将下载文件输出到stdout□将进度信息输出到stderr□不显示进度信息使用`--silent`选项。

```
curl URL --silent
```

这条命令是将下载文件输出到终端，所有下载的数据都被写入到stdout□

使用选项`-O`将下载的数据写入到文件，必须使用文件的绝对地址：

```
curl http://example.com/text.iso --silent -O
```

选项-o将下载数据写入到指定名称的文件中，并使用--progress显示进度条：

```
curl http://example.com/test.iso -o filename.iso --progress
######################################## 100.0%
```

不输出错误和进度信息

-s 参数将不输出错误和进度信息。

```
curl -s https://www.example.com
# 上面命令一旦发生错误，不会显示错误信息。不发生错误的话，会正常显示运行结果。
```

如果想让 curl 不产生任何输出，可以使用下面的命令。

```
curl -s -o /dev/null https://google.com
```

断点续传

curl能够从特定的文件偏移处继续下载，它可以通过指定一个便宜量来下载部分文件：

```
curl URL/File -C 偏移量

#偏移量是以字节为单位的整数，如果让curl自动推断出正确的续传位置使用-C -
curl -C -URL
```

使用curl设置参照页字符串

参照页是位于HTTP头部中的一个字符串，用来表示用户是从哪个页面到达当前页面的，如果用户点击网页A中的某个连接，那么用户就会跳转到B网页，网页B头部的参照页字符串就包含网页A的URL

使用--referer选项指定参照页字符串：

```
curl --referer http://www.google.com http://wangchujiang.com
```

用curl设置用户代理字符串

有些网站访问会提示只能使用IE浏览器来访问，这是因为这些网站设置了检查用户代理，可以使用curl把用户代理设置为IE这样就可以访问了。使用--user-agent或者-A选项：

```
curl URL --user-agent "Mozilla/5.0"
curl URL -A "Mozilla/5.0"
```

其他HTTP头部信息也可以使用curl来发送，使用-H"头部信息"传递多个头部信息，例如：

```
curl -H "Host:wangchujiang.com" -H "accept-language:zh-cn" URL
```

curl的带宽控制和下载配额

使用--limit-rate限制curl的下载速度：

```
curl URL --limit-rate 50k
```

命令中用k□千字节）和m□兆字节）指定下载速度限制。

使用--max-filesize指定可下载的最大文件大小：

```
curl URL --max-filesize bytes
```

如果文件大小超出限制，命令则返回一个非0退出码，如果命令正常则返回0。

```
curl --limit-rate 200k https://example.com
# 上面命令将带宽限制在每秒 200K 字节。
```

用curl进行认证

使用curl选项-u可以完成HTTP或者FTP的认证，可以指定密码，也可以不指定密码在后续操作中输入密码：

```
curl -u user:pwd http://wangchujiang.com
curl -u user http://wangchujiang.com
```

只打印响应头部信息

通过-I或者-head可以只打印出HTTP头部信息：

```
[root@localhost text]# curl -I http://wangchujiang.com
HTTP/1.1 200 OK
Server: nginx/1.2.5
date: Mon, 10 Dec 2012 09:24:34 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Vary: Accept-Encoding
X-Pingback: http://wangchujiang.com/xmlrpc.php
```

get请求

```
curl "http://www.wangchujiang.com"    # 如果这里的URL指向的是一个文件或者一幅图都可以直接下载到本地
curl -i "http://www.wangchujiang.com" # 显示全部信息
curl -l "http://www.wangchujiang.com" # 只显示头部信息
curl -v "http://www.wangchujiang.com" # 显示get请求全过程解析
```

post请求

```
$ curl -d "param1=value1&param2=value2" "http://www.wangchujiang.com/login"

curl -d'login=emma□password=123' -X POST https://wangchujiang.com/login
# 或者
$ curl -d 'login=emma' -d 'password=123' -X POST
https://wangchujiang.com/login
```

--data-urlencode 参数等同于 -d，发送 POST 请求的数据体，区别在于会自动将发送的数据进行 URL 编码。

```
curl --data-urlencode 'comment=hello world' https://wangchujiang.com/login
# 上面代码中，发送的数据hello world之间有一个空格，需要进行 URL 编码。
```

读取本地文本文件的数据，向服务器发送

```
curl -d '@data.txt' https://wangchujiang.com/upload
# 读取data.txt文件的内容，作为数据体向服务器发送。
```

json格式的post请求

```
curl -l -H "Content-type: application/json" -X POST -d
'{"phone":"13521389587","password":"test"}'
http://wangchujiang.com/apis/users.json
```

向服务器发送 Cookie

使用--cookie "COKKIES"选项来指定cookie，多个cookie使用分号分隔：

```
curl http://wangchujiang.com --cookie "user=root;pass=123456"
```

将cookie另存为一个文件，使用--cookie-jar选项：

```
curl URL --cookie-jar cookie_file
```

-b 参数用来向服务器发送 Cookie。

```
curl -b 'foo=bar' https://taobao.com
# 上面命令会生成一个标头Cookie: foo=bar，向服务器发送一个名为foo，值为bar的 Cookie。
```

```
curl -b 'foo1=bar' -b 'foo2=baz' https://taobao.com
# 上面命令发送两个 Cookie。
```

curl -b cookies.txt https://www.taobao.com

上面命令读取本地文件 cookies.txt，里面是服务器设置的 Cookie（参见-c参数），将其发送到服务器。

```
**Cookie 写入一个文件**
```

curl -c cookies.txt https://www.taobao.com

上面命令将服务器的 HTTP 回应所设置 Cookie 写入文本文件cookies.txt。

```
**请求的来源**
```

`-e` 参数用来设置 `HTTP` 的标头 `Referer`，表示请求的来源。

curl -e 'https://taobao.com?q=example' https://www.example.com

上面命令将Referer标头设为

`-H` 参数可以通过直接添加标头 `Referer`，达到同样效果。

curl -H 'Referer: https://taobao.com?q=example' https://www.example.com

上传二进制文件

`-F` 参数用来向服务器上传二进制文件。

curl -F 'file=@photo.png' https://taobao.com/profile

上面命令会给 HTTP 请求加上标头 Content-Type: multipart/form-data，然后将文件photo.png作为file字段上传。

`-F` 参数可以指定 `MIME` 类型。

curl -F 'file=@photo.png;type=image/png' https://taobao.com/profile

上面命令指定 MIME 类型为image/png，否则 curl 会把 MIME 类型设为 application/octet-stream。

`-F` 参数也可以指定文件名。

curl -F 'file=@photo.png;filename=me.png' https://taobao.com/profile

上面命令中，原始文件名为photo.png，但是服务器接收到的文件名为me.png。

设置请求头

`-H` 参数添加 `HTTP` 请求的标头。

curl -H 'Accept-Language: en-US' https://google.com

上面命令添加 HTTP 标头 Accept-Language: en-US。

curl -H 'Accept-Language: en-US' -H 'Secret-Message: xyzzy' https://google.com

上面命令添加两个 HTTP 标头。

curl -d '{"login": "emma", "pass": "123"}' -H 'Content-Type: application/json' https://google.com/login

上面命令添加 HTTP 请求的标头是 Content-Type: application/json，然后用 -d 参数发送 JSON 数据。

跳过 SSL 检测

curl -k https://www.example.com

上面命令不会检查服务器的 SSL 证书是否正确。

请求跟随服务器的重定向

`-L` 参数会让 `HTTP` 请求跟随服务器的重定向，`curl` 默认不跟随重定向。

curl -L -d 'tweet=hi' https://api.example.com/tweet

调试参数

`-v` 参数输出通信的整个过程，用于调试。

curl -v https://www.example.com

--trace参数也可以用于调试，还会输出原始的二进制数据。

$ curl --trace - https://www.example.com

获取本机外网ip

curl ipecho.net/plain

使用 curl 测试网站加载速度

命令有一个鲜为人知的选项`-w`该选项在请求结束之后打印本次请求的统计数据到标准输出。

首先，我们定义控制打印行为的格式化字符串。新建文本文件 `fmt.txt`并填入下面的内容：

\n Response Time for: %{url_effective}\n\n DNS Lookup Time:\t\t%{time_namelookup}s\n Redirection Time:\t\t%{time_redirect}s\n Connection Time:\t\t%{time_connect}s\n App Connection Time:\t\t%{time_appconnect}s\n Pre-transfer Time:\t\t%{time_pretransfer}s\n Start-transfer Time:\t\t%{time_starttransfer}s\n\n Total Time:\t\t\t%{time_total}s\n

curl 提供了很多置换变量，可以在格式化字符串中通过 `%{var}` 的形式使用。完整的变量列表可以在 `curl` 的`manpage` 中查看。简单介绍一下我们使用的这几个变量：

- `url_effective`: 执行完地址重定向之后的最终 URL
- `time_namelookup`: 从请求开始至完成名称解析所花的时间，单位为秒，下同；
- `time_redirect`: 执行所有重定向所花的时间；
- `time_connect`: 从请求开始至建立 TCP 连接所花的时间；
- `time_appconnect`: 从请求开始至完成 SSL/SSH 握手所花的时间；
- `time_pretransfer`: 从请求开始至服务器准备传送文件所花的时间，包含了传送协商时间；
- `time_starttransfer`: 从请求开始至服务器准备传送第一个字节所花的时间；
- `time_total`: 完整耗时。

然后执行请求，通过 @filename 指定保存了格式化字符串的文件：

$ curl -L -s -w @fmt.txt -o /dev/null http://www.google.com

输出：

Response Time for: http://www.google.co.jp/?gfe_rd=cr&dcr=0&ei=cjIaWpTkHeiQ8QfnxYzoBA

DNS Lookup Time: 0.000038s Redirection Time: 0.207271s Connection Time: 0.000039s App Connection Time: 0.000039s Pre-transfer Time: 0.000067s Start-transfer Time: 0.260115s

Total Time: 0.467691s

要求返回是压缩的状态

▶ curl --compressed -o- -L https://yarnpkg.com/install.sh | bash % Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 54 100 54 0 0 42 0 0:00:01 0:00:01 --:--:-- 42 100 2341 100 2341 0 0 1202 0 0:00:01 0:00:01 --:--:-- 9289 Installing Yarn!

> Downloading tarball...

[1/2]: https://yarnpkg.com/latest.tar.gz -->

/var/folders/j7/3xly5sk567s65ny5dnr__3b80000gn/T/yarn.tar.gz.XXXXXXXXXX.9hJsBsrA % Total % Received % Xferd Average Speed Time Time Time Current

```
                                 Dload  Upload   Total   Spent    Left  Speed
```

100 57 100 57 0 0 72 0 --:--:-- --:--:-- --:--:-- 72 100 93 100 93 0 0 63 0 0:00:01 0:00:01 --:--:-- 63 100 643 100 643 0 0 248 0 0:00:02 0:00:02 --:--:-- 707 100 1215k 100 1215k 0 0 153k 0 0:00:07 0:00:07 --:--:-- 305k

[2/2]: https://yarnpkg.com/latest.tar.gz.asc --> /var/folders/j7/3xly5sk567s65ny5dnr__3b80000gn/T/yarn.tar.gz.XXXXXXXXXX.9hJsBsrA.asc 100 61 100 61 0 0 356 0 --:--:-- --:--:-- --:--:-- 356 100 97 100 97 0 0 325 0 --:--:-- --:--:-- --:--:-- 325 100 647 100 647 0 0 1283 0 --:--:-- --:--:-- --:--:-- 1283 100 832 100 832 0 0 1107 0 --:--:-- --:--:-- --:--:-- 812k