cURL.1 手册页

名称

cURL - transfer a URL

摘要

cURL[选项] [URL...]

描述

cURL 是一个向服务器或从服务器传输数据的工具，它支持 HTTP 、HTTPS、FTP 、 FTPS 、 SCP、SFTP、TFTP、DICT、TELNET、LDAP 或 FILE 等协议。该命令设计为无需用户干预即可执行。
cURL 提供了一大堆诸如代理支持、用户认证、FTP 上传、HTTP POST、SSL 连接、Cookies、文件续传等等有用的技巧。正如你将在下面看到的，如此多的特性会让你头晕目眩！
cURL 的所有传输相关特性的是由 libcURL 所支持。详情见 libcURL (3)。

URL：

URL 语法是协议相关的。您可在 RFC 3986找到详细解释。
您可以指定多个 URL 或在大括号{}内写入 URL 的一部分形成 URL 的集合：
http://site.{one,two,three}.com
或者您也可以使用[ ]获得连续的字母或数字序列：
ftp://ftp.numericals.com/file[1-100].txt ftp://ftp.numericals.com/file[001-100].txt （前面带有0的) ftp://ftp.letters.com/file[a-z].txt
目前还不支持嵌套的序列，但是您可以在一条 URL 中混合使用多个序列：
http://any.org/archive[1996-1999]/vol[1-4]/part{a,b,c}.html
您可在命令行中指定任意数量的 URL。它们将以指定的顺序被逐一获取。
从 cURL 7.15.1版以来，您也可以指定步长范围，如此您就可以得到所有带第 N 个数字或字母的 URL：
http://www.numericals.com/file[1-100:10].txt http://www.letters.com/file[a-z:2].txt
如果您指定的 URL 没有协议://前缀，cURL 会尝试猜测您可能需要的协议。默认使用 HTTP 协议，但会基于常用主机名前缀而尝试使用其它协议。例如，对于以“FTP”开头的主机名称，cURL 将假定使用 FTP 协议 。
cURL 会尝试为多个文件的传输重用连接，从而使从同一台服务器上获取文件不会做多重连接/握手。这种方法提高了速度。当然，这只对在同一命令行中指定的文件有效，而且不能用于单独的 cURL 之间的调用。

进度指示器

cURL 在操作时通常会显示一个进度指示器，显示已传输的数据量、传输速度和估计剩余时间等。
然而，由于 cURL 默认在终端上显示此数据，如果您调用 cURL 去做一个向终端写入数据的操作，它将禁用进度指示器，否则它会将进度指示信息与输出的返回数据混淆在一起。
如果您需要对应 HTTP POST 或 PUT 请求的进度指示器，您得使用 shell 的重定向操作符(>)、 -o [文件名]或其他类似的操作，将返回的数据重定向到文件中。这与 FTP 上传操作不同，该操作不会向终端输出任何响应数据。
如果您想用进度“条”代替默认的显示， -#选项会很有帮助。

选项

对于所有的布尔选项(option)，使用 --option 来启用，使用 --no-option 来禁用。也就是说，您使用完全相同的选项名称，但须加上前缀“no-”然而，在此列表中，我们通常只会列出并显示它们的--option。(--no-option 这个概念是在7.19.0版中添加的。以前大多数选项的开启/关闭是重复使用相同的命令行选项。)
-a/--append
(FTP/SFTP)当在上传操作中使用时，此选项将告诉 cURL 将内容附加到目标文件，而不是覆盖它。如果该文件不存在，将创建一个新文件。请注意，此标志将被一些 SSH 服务器所忽略(包括 OpenSSH) 。
-A/--user-agent <用户代理字符串>
(HTTP)指定用户代理(User-Agent)字符串发送给 HTTP 服务器。如果这里没有设置为“ Mozilla/4.0 ”，一些设计的不好的 CGI 程序将不能正常工作。如字符串包含空白字符，则用单引号括出此字符串。此选项也可用-H/--header 来设置。
如果此选项设置超过一次，将使用最近一次的设置。
--anyauth
(HTTP) 告知 cURL 自己找出远程站点声明支持的最安全的验证方法并使用之。由发送一个请求并检查 response-headers 来实现，从而可能诱发额外的网络流量。这是用来代替具体验证方法设定，您可配合--basic、--digest 、--ntlm、与--nagotiate 使用。
请注意，如果您从标准输入(stdin)上传，则不推荐使用--anyauth，因为它可能需要将数据发送两次而且届时客户端必须支持回卷(rewind)。如从标准输入(stdin)上传时出现此需求，上传操作将失败。
-b/--cookie <name=data>
(HTTP)将数据作为 cookie 传递给 HTTP 服务器。这些数据可能是前次从服务器 Set-Cookie 行返回的。这些数据应是“NAME1=VALUE1; NAME2=VALUE2”这样的格式。
如果在行中没有使用'='符号，那么它将被视为一个保存了前次 cookie 行的文件名，如果此文件与前次保存的 cookie 行匹配，则它在此会话中将被读取。使用这种方法也激活会激活 "cookie 解析器" ，它将使 cURL 同时记录传入的 cookie，这样可方便的与-L/--location 选项结合使用。被读取 cookie 的文件格式应为纯文本的 HTTP headers 或 Netscape/Mozilla 的 cookie 文件格式。

请注意，被-b/--cookie 指定的文件只能作为输入使用，没有 cookies 会被存储在文件中。为了存储 cookie，用-c/--cookie-jar 选项或您甚至可以用-D/--dump-header 选项把 HTTP header 保存到文件中！
如果此选项设置超过一次，将使用最近一次的设置。
-B/--use-ascii
当使用 FTP 或 LDAP 时启用 ASCII 传输。使用 FTP 时，也可以使用一个以";type=A"结尾的 URL 来强制执行此选项。此选项导致数据以文本模式发送到Win32系统的标准输出(stdout)。
--basic
(HTTP)告知 cURL 使用 HTTP 基本验证。这是默认选项，并且该选项通常是无意义的，除非你用它来覆盖先前设置的不同的身份验证方法的选项，(如--ntlm 、--digest 、或--negotiate) 。
--ciphers <加密方式列表>
(SSL)具体指定在连接中使用哪些加密方式。列表必须是有效的加密方式。在此网址阅读 SSL 加密方式列表的细节：http://www.openssl.org/docs/apps/ciphers.html
NSS 加密方式与 openssl 和 GnuTLS 不同。NSS 加密方式的完整列表在此网址：http://directory.fedora.redhat.com/docs/mod_nss.html#Directives 的 NSSCipherSuite 条目中。
假如此选项已被使用过多次，则最后一次的设置将取代其它的。
--compressed
(HTTP)使用 libcurl 支持的一种压缩算法请求一个压缩的响应，并返回解压缩的文档。如果使用此选项并且服务器发送了一个不受支持的压缩方式，cURL 将报错。
--connect-timeout <秒>
您连接到服务器所允许花费的最大时间，单位为秒。只限于连接阶段，一旦 crul 已经连接，该选项就不起作用了。又见-m/--max-time 选项。
假如此选项已被使用过多次，则使用最后一次的设置。
-c/--cookie-jar <文件名>
指定您想要 cURL 完成操作后将所有 cookie 写入哪个文件中。cURL 将写入之前从指定文件中读取的、以及从远程服务器获取的所有 cookie。如果无 cookie 信息，则不会写入文件。该文件将使用 Netscape 的 cookie 文件格式。如果您设置文件名为半角单破折号"-"，这些 cookies 信息会被写入标准输出(stdout)。
注意：如果 Cookie jar 无法被建立或写入，整个 cURL 操作不会失败甚至也不会报错。使用-v 将得到一个警告显示，但这将是在可能发生致命错误时您唯一可见的反馈。
假如此选项已使用过多次，则将使用最后一次指定的文件名。
-C/--continue-at <偏移地址>
在指定偏移地址继续/恢复之前的文件传输。给定的偏移地址是一个将被忽略的确切字节数，在源文件传输到目的地之前从文件起始处计算。如果在上传中使用，FTP 服务器的 SIZE 命令将不能用于 cURL。
使用“ -C - ”来告知 cURL 在何处及如何自动恢复传输。然后使用给定的输出/输入文件来做到这一点。
假如此选项已被使用过多次，则使用最后一次的设置。

--create-dirs
配合-o 选项使用时，cURL 将根据需要建立必要的本地目录结构。此选项会建立在-o 选项中提到的目录，除此以外没有其它用途。如果-o 选项中涉及的文件名没有使用目录或目录已存在，则不会建立目录。
要使用 FTP 或 SFTP 创建远程目录时，请使用--ftp-create-dirs 选项。
--crlf
(FTP)在上传时将 LF 转换为 CRLF。在 MVS 中有用(OS/390) 。
-d/--data <数据>
(HTTP)将特定数据 POST 到服务器，与在浏览器中用户填写表单并按提交按钮的效果一样。cURL 使用 mime type content-type application/x-www-form-urlencoded 传输数据到服务器。请与-F/--form 比较。
-d/--data 与--data-ascii 相同 。如果要 post 二进制数据，您应该使用--data-binary 选项。如果要 post URL 编码的表单项，您应该采用--data-urlencode 。
在同一个命令行多次使用该参数，数据块将被&符号合并提交。因此，使用'-d name=daniel -d skill=lousy'后提交的数据是'name=daniel&skill=lousy' 。
如果您的数据以字符@开头，其余的部分应当是一个用来读取数据的文件名，或者您想要 cURL 从标准输入(stdin)读取数据。该文件的内容必须是经过 URL 编码的。也可指定多个文件。比如要 post 文件 foobar，那么格式是--data @foobar 。
--data-binary <数据>
(HTTP)post 二进制数据。
如果您的数据以字符@开头，其余的部分应当是个文件名。post 的数据使用 --data-ascii 类似的方式，区别是换行是保留的，未作转换。
如果此选项被多次使用，则第一次定义的数据会附加到-d/--data 所描述的数据后 。
--data-urlencode <数据>
(HTTP)与 --data 选项相同，唯一的差别是要采用 url 编码。(在7.18.0版加入)
要做 CGI 兼容，<数据>部分应由 name 以及其后的分隔符和指定内容组成。该<数据>部分使用下列语法：
content
将使 cURL 使用 url-encode 编码内容并传递。只是要小心，这样的内容不包含任何=或@符号，所以这将会使语法与下面其他情况相匹配。
=content
将使 cURL 使用 url-encode 编码内容并传递。前面的=号不包含在数据中。
name=content
将使 cURL 使用 url-encode 编码内容部分并传递。请注意，name 的部分已经预先经过 URL-encoded 编码了。
@filename
将使 cURL 从给定的文件(包含任何换行符的)中读取数据，URL-encode 编码数据并在 post 时传递给它。
name@filename
将使 cURL 从给定的文件(包含任何换行符的)中读取数据，URL-encode 编码数据并在 post 时传递给它。name 部分获得相同的附加，致使 name=urlencoded-file-content。请注意，name 已经预先经过 URL-encoded 编码了。
--digest

(HTTP)允许 HTTP Digest 认证。这是一个避免密码以明文传送的认证。使用此选项与-u/--user 选项组合来设置用户名和密码。又见--ntlm 、--negotiate 和--anyauth 相关选项。
如果多次使用该选项，下列事件不作任何区别。
--disable-eprt
(FTP)告知 cURL 在做主动 FTP 传输时禁用 ERPT 和 LPRT 命令。cURL 在使用 PORT 命令前，通常会先尝试使用 EPRT ，然后 LPRT，但使用此选项后，就会先使用 PORT 命令了。EPRT 和 LPRT 是原 FTP 协议的扩展，并可能无法在所有的服务器上工作，但它们以更好的方式拥有比传统的 PORT 命令更多的功能。
自 cURL7.19.0起，--eprt 可以用来再次启用 EPRT，--no-eprt 是--disable-eprt 的别名 。
禁用 EPRT 仅改变主动行为。如果你想切换到被动模式(passive mode)请您不要使用-P/--ftp-port 或强制与--ftp-pasv 一起使用。
--disable-epsv
(FTP)告知 cURL 在做被动 FTP 传输时禁用 EPSV 命令。cURL 在使用 PASV 命令前，通常会先尝试使用 EPSV，但使用此选项后，就不会先尝试 EPSV 命令了。
自 cURL7.19.0起，--epsv 可以用来再次启用 EPSV，--no-epsv 是--disable-epsv 的别名 。
禁用 EPSV 仅改变被动的行为。如果想切换到主动模式下您需要使用-P/--ftp-port 。
-D/--dump-header <文件>
将协议头写入指定文件。
此选项是方便您保存 HTTP 网站发送给您的 HTTP header 时用的。保存协议头的 cookie 数据就可被第二个 cURL 通过-b/--cookie 选项调用！然而-c/--cookie-jar 选项还是更好的储存 cookie 的方法。
当使用 FTP 时，FTP 服务器的应答行被当作是“协议头”，从而被保存。
假如此选项已被使用过多次，则使用最后一次的设置。
-e/--referer <URL>
(HTTP)向 HTTP 服务器传送“参考页”的信息。此选项当然也可以与-H/--header 标记一起使用。当与-L/--location 一起使用时，您可以附加“;auto”到--referer URL，使 curl 自动设置上一个 URL，当它后面跟随 Location: header 时。";auto"字符串可以单独使用，即使您没有设置初始的--referer。
假如此选项已被使用过多次，则使用最后一次的设置。
--engine <name>
为加密操作选择 openssl 加密引擎。使用--engine list 打印编译时所支持引擎的清单。请注意，并非所有(或没有)引擎可用在运行时。
--environment
(仅在 RISC 操作系统)设置了一系列的环境变量，使用-w 选项支持的名称，允许运行 cURL 后更容易的提取有用的信息。
--egd-file <文件>
(SSL)为 Entropy Gathering Daemon 套接字指定路径名。此套接字用于为 SSL 连接产生随机种子。又见 --random-file 的选项。
-E/--cert <证书[:密码]>
(SSL)告知 cURL 当通过 HTTPS 或 FTPS 获取文件时使用指定的证书文件。证

书必须是 PEM 格式。如果没有指定可选密码，将在终端上提示输入。注意，此选项假设的“证书”文件，是私钥和私人证书连结！见--cert 和--key 单独指定他们。
如果 curl 使用 NSS SSL 库编译，那么此选项告知 curl 用在由环境变量 SSL_DIR(或默认的/etc/pki/nssdb)定义的 NSS 数据库中的证书别名。如果 NSS PEM PKCS#11 模块(libnsspem.so)是可用的，那么将加载 PEM 文件。
假如此选项已被使用过多次，则使用最后一次的设置。
--cert-type <类型>
(SSL)告知 curl 所提供证书的类型。PEM、DER 和 ENG 是可被识别的类型。如果没有指定，将假定为 PEM。
假如此选项已被使用过多次，则使用最后一次的设置。
- cacert <CA certificate>
(SSL)告知 curl 使用指定的证书文件以验证 peer。该文件可能包含多个 CA 证书。证书必须为 PEM 格式。通常 curl 使用内置的默认文件，所以此选项通常用于改变默认的文件。
curl 确认已设置的名为'cURL_CA_BUNDLE'的环境变量，并使用给定的路径作为一个路径捆绑的 CA 证书。此选项会覆写该变量。
Windows 版本的 curl 会自动寻找文件名为'curl-ca-bundle.crt' 的 CA 证书，无论是在 cURL.exe 同一目录下，或在当前工作目录下，或在任何文件夹您的 PATH 中的任意文件夹下。
如果 curl 使用 NSS SSL 库编译，那么此选项告知 curl 用在由环境变量 SSL_DIR(或默认的/etc/pki/nssdb)定义的 NSS 数据库中的证书别名。如果 NSS PEM PKCS#11 模块(libnsspem.so)是可用的，那么将加载 PEM 文件。
假如此选项已被使用过多次，则使用最后一次的设置。
--capath <CA 证书目录>
(SSL)告知 curl 使用指定的证书目录以验证 peer。该证书必须是 PEM 格式的，而且目录必须已使用 OpenSSL 提供的 c_rehash 工具提供处理过。如果--cacert 指定的文件包含许多 CA 证书，使用--capath 可以让 curl 使 SSL 连接效率远远高于使用--cacert。
假如此选项已被使用过多次，则使用最后一次的设置。
-f/--fail
(HTTP)服务器出错时屏蔽错误信息(无任何输出)。这样做以便更好的支持脚本等能更好的处理失败的尝试。在正常情况下当一个 HTTP 服务器无法提供文档时，它将返回一个解释其原因的 HTML 文件 。此标志将防止 curl 输出此文档，并返回错误22 。
这种方法不是万无一失的，有非成功响应代码通过的场合，尤其是参与验证时(响应代码401和407) 。
--ftp-account [数据]
(FTP)当 FTP 服务器在提供用户名和密码后请求“帐户数据”，此数据通过 ACCT 命令发送。(在7.13.0 版加入)
如果该选项使用过两次，则第二次将覆盖上一次的使用。
--ftp-create-dirs
(FTP / SFTP)当 FTP 或 SFTP 的 URL/操作使用了一个当前不存在于服务器上的路径时，标准的 crul 行为会失败。使用此选项，curl 将尝试创建丢失的目录。
--ftp-method [方法]

(FTP)控制 curl 应使用的获取 FTP 服务器中文件的方法。该方法的变量应是以下的选项：
multicwd
curl 为每一个给定 URL 的路径运行一个单一的 CWD 操作。目录结构复杂的话这将意味着很多条命令。这是 RFC1738所描述的。这是预设的，但最慢的行为。
nocwd
curl 完全执行 CWD 操作，而是使用 SIZE、RETR、STOR 等命令并向服务器发送所有这些命令的完整的路径。这是最快的行为。
singlecwd
curl 为完整的目标目录做一次 CWD 操作，然后“正常地”操作文件(如在 multicwd 中一样)。这比' nocwd '符合标准，但又比' multicwd '开销要小。
(在7.15.1版加入)
--ftp-pasv
(FTP)使用被动模式的数据连接。被动模式是内部默认的行为，但使用此选项可以用来覆盖上一个-P/--ftp-port 选项。(在7.11.0 版加入)
如果此选项被多次使用，下列事件不作任何区别。取消强制被动模式是不可行的，但您届时必须强制再次以正确的-P/--ftp-port 取代。
被动模式意味着 curl 将先尝试 EPSV 命令，然后执行 PASV 命令，除非使用了--disable-epsv 选项。
--ftp-alternative-to-user <命令>
(FTP)如果使用 USER 和 PASS 命令认证失败，则发送此命令。当使用客户端证书链接到 FTPS 上的 Tumbleweed 的安全传输服务器时，使用“SITE AUTH”将告知服务器从证书检索用户名。(在7.15.5版加入)
--ftp-skip-pasv-ip
(FTP)在 curl 连接到数据连接时，告知 cURL 不要使用服务器在响应 curl 的 PASV 命令时建议的 IP 地址。相反 curl 将重用在控制连接时使用的相同的 IP 地址。(在7.14.2版加入)
如果 PORT、EPRT 或 EPSV 是用来代替 PASV 的话，此选项无效。
--ftp-ssl
(FTP)尝试使用 SSL/TLS 的 FTP 连接。如果服务器不支持 SSL/TLS 则恢复到一个非安全连接。为不同级别的加密需求又见--ftp-ssl-control 和--ftp-ssl-reqd。(在7.11.0 版加入)
--ftp-ssl-control
(FTP)请求 SSL/TLS 的 FTP 登录，不加密传输。允许安全认证，但为了传输效率不加密数据传输。如果服务器不支持 SSL / TLS 的则传输失败。 (在7.16.0版加入)
--ftp-ssl-reqd
(FTP)请求 SSL / TLS 的 FTP 连接。如果服务器不支持 SSL / TLS 的则终止连接。(在7.15.5版加入)
--ftp-ssl-ccc
(FTP)验证后使用 CCC(Clear Command Channel)关闭 SSL / TLS 层。其余的控制通道通信将不加密。此操作允许 NAT 路由跟随 FTP 事务。默认为被动模式。选项的其他方式见 --ftp-ssl-ccc-mode 。(在7.16.1 版加入)
--ftp-ssl-ccc-mode [active/passive]

(FTP)使用 CCC(Clear Command Channel)设置 CCC 模式。被动模式不接受 shutdown 命令，而是等待服务器这样做，也不会回应服务器的 shutdown 命令。主动模式接受 shutdown 命令并等待服务器回应。(在7.16.2版加入)
-F/--form <name=content>
(HTTP)curl 模拟填写表格并按下提交按钮。这将使 curl 根据 RFC1867使用 Content-Type multipart/form-data 来 POST 数据。这将使二进制文件等得以上传。强制'content'部分为文件，在文件名前加上前缀@号。如只需从文件中得到content部分的内容，在文件名前使用前缀<。@与<的区别在与，@让一个文件在 POST 动作中作为文件上传，而<则仅从文件中获得内容作为文本域。
例如，如果要发送密码文件到服务器，'password'是表单域的名称，这里将填入 /etc/passwd 文件：
curl -F password=@/etc/passwd www.mypasswords.com
如果要在标准输入(stdin)中获取输入来代替文件，则在应提供文件名的位置上使用-代替。这同样适用于@和<的结构。
您也可以使用';type='告知 curl 是什么 Content-Type，类似于：
curl -F "web=@index.html;type=text/html" url.com
或是：
curl -F "name=daniel;type=text/foo" url.com
您也可以设定'filename='来明确地变更名称域的一个文件上传部分 ，就像这样：
curl -F "file=@localfile;filename=nameinpost" url.com
更多的实例和细节见手册。
此选项可以多次使用。
--form-string <name=string>
(HTTP)类似--form，除了命名参数的值字符串是字面上的意义。以' @ '和' < '开头字符以及值中的';type='字符串没有特殊的含义。如果字符串值可能碰巧触发 --form 的' @ '或' < '特性，则在--form 中优先使用此参数。
-g/--globoff
此选项关闭"URL 通配符分析器" 。当设定此选项，您可以指定包含{}[]字符的的 URL，而不被 curl 本身解释。请注意，这些字符不是普通合法的 URL 内容，但应根据 URI 的标准编码。
-G/--get
此选项将所有-d/--data 或--data-binary 的指定数据被用在 HTTP GET 请求而不是 POST 请求，否则将使用 POST。这些数据与'?'分隔符附加在 URL 中。
如果结合-I 使用，POST 数据将取代并与 HEAD 请求一起附加到 URL 中。
如果此选项被多次使用，下列事件不作任何区别。这是因为撤消 GET 是没有道理的，但是您应当选择您较喜欢的方法来取代。
-h/--help
使用帮助。
-H/--header <header>
(HTTP)在获取网页时使用额外的 header。您可以指定任何数量的额外的 header。请注意，如果您要添加与 curl 内部使用的 header 名称相同的自定义 header，您的外部 header 设置将取代内定的。这样，您就能做比 curl 一般能做到的更多的事。假如您不完全了解您在做的事，那么就不应当取代内部设定的 header。在冒号右侧给出一个无内容的替换来移除一个内部的 header，如： -H "主机: " 。

cURL 将确保您添加 / 取代的每一个 header 都与正常的行尾标记一同发送，因此您不应把它作为作为 header 的内容添加进来：不添加换行符或回车，它们只会把您的东西弄得一团糟。
又见-A/--user-agent 和-e/--referer 选项。
此选项可多次使用来添加 / 替换 / 移除多个 header。
- hostpubmd5 <md5>
传递一个包含32个十六进制数字的字符串。该字符串应是远程主机公钥的128位MD5校验，除非 md5sums 匹配，否则 cURL 将拒绝连接。此选项仅用于 SCP 和 SFTP 传输。(在7.17.1版中加入)
--ignore-content-length
(HTTP)忽略 Content-Length header。运行 Apache1.x 的服务器对大于2G 字节的文件会报告 Content-Length 不正确，此选项对此很有用。
-i/--include
(HTTP)在输出中包含的 HTTP－header。HTTP－header 中包含了像是服务器名称、文档日期、HTTP 协议的版本和更多信息...
--interface<名称>
使用指定界面执行操作。您可以输入界面名称， IP 地址或主机名。一个例子可能如下所示：
curl --interface eth0:1 http://www.netscape.com/
假如此选项已被使用过多次，则使用最后一次的设置。
-I/--head
(的 HTTP / FTP /档案)只获取 HTTP－header！HTTP 服务器功能命令 HEAD，使用此选项只获取文档的 header。当使用的 FTP 或 FILE 文件，cURL 只显示文件大小和最后修改时间。
-j/--junk-session-cookies
(HTTP)当 curl 被告知给定文件中读取 cookie 时，此选项会放弃所有“会话 cookie ”。这将基本上与开始一个新会话起到相同的效果。典型的浏览器关闭时总是舍弃会话 cookie 的。
-k/--insecure
(SSL)此选项明确允许 cURL 执行“不安全” SSL 连接和传输。所有的 SSL 连接使用捆绑默认安装的 CA 证书试图使其安全。这使得除非使用-k/--insecure，否则所有连接被认为是“不安全”的而失败。
进一步的细节请看网上资源： http://cURL.haxx.se/docs/sslcerts.html
--keepalive-time <秒>
此选项设定一个连接在发送保持活动探针之前的空闲时间和独立的保持活动探针的间隔。这是当前在操作系统提供的 TCP_KEEPIDLE 和 TCP_KEEPINTVL 套接字选项有效的(意味着 Linux、AIX、HP-UX 等等)。如果使用--no-keepalive 选项，则此选项不发生效力。(在7.18.0版加入)
如果多次使用此选项，则最后一次的设定起作用。
--key <私钥>
(SSL / SSH)私钥文件名。允许您提供在此单独的文件中提供您的私钥。
假如此选项已被使用过多次，则使用最后一次的设置。
--key-type <类型>
(SSL)私钥文件类型。指定--key 输入的私钥类型。支持的类型有 DER、PEM 和

ENG。如果没有指定，将假定为 PEM。
假如此选项已被使用过多次，则使用最后一次的设置。
--krb <安全等级>
(FTP)启用并使用 Kerberos 身份验证。必须输入安全等级，应该是'clear' 、'safe' 、'confidential'、或'private' 。如果您指定的安全等级不在以上几种以内，则将使用'private'代替。
此选项需要有在 berberos4或 GSSAPI(GSS-Nagotiate)支持下编译的库文件。这并不是很常见。使用-V/--version ，看看您的 cURL 是否支持。
假如此选项已被使用过多次，则使用最后一次的设置。
-K/--config <配置文件>
指定用来读取 cURL 参数的配置文件。此配置文件是一个保存了命令行参数的文本文件，使用时就像参数被书写在实际的命令行中一样。选项及其参数必须在同一配置文件的行中，由空格、冒号、等号或其他连接符号组成(但首选分隔符是等号)。如果该参数中包含空格，则必须使用引号括出。在双引号中可用以下转义符号：\\, \", \t, \n, \r 与 \v。在其它任何字母前的反斜杠将被忽略。 如果配置文件行首以' # '字符开始，则此行的剩余部分将被视为注释。在配置文件中，每行请只写入一个选项。
如为-K/--config 指定文件名为' - ' 则 cURL 会从标准输入(stdin)获取文件名 。
请注意，您需要使用--url 选项才能在配置文件中指定一个 URL，而不是简单的在写上一个 URL。因此，它可能与此类似：
url= “ http://cURL.haxx.se/docs/ ”
长选项名可在配置文件中以没有开头的双破折号"--"的形式存在。
当 cURL 被引用时，它总是(除非使用-q 选项)检查是否有默认的配置文件，如有则使用。将在下列位置按顺序查找默认的配置文件：
1)cURL 曲试图找到“HOME 目录”：它首先检查 cURL_HOME ，然后在 HOME 环境变量中查找。如果做不到这一点，它会在类 UNIX 系统中使用 getpwuid()(返回当前系统用户的 HOME 目录) 。在 Windows 中，它会检查 APPDATA 变量或检查'%USERPROFILE%\Application Data'以作为最后的手段。
2)在 Windows 中，如果 HOME 目录中没有_curlrc 文件，它将检查 cURL 可执行文件所在目录。在类 UNIX 系统中，它只是尝试从确定的 HOME 目录里加载.curlrc。

```
# --- 示例文件 --- #这是一个注释 url = "curl.haxx.se" output = "curlhere.html" user-agent = "superagent/1.0"

# 攫取另一个 URL url = "curl.haxx.se/docs/manpage.html" -O referer = "http://nowhereatall.com/" #- -- 示例文件结束 ---
```

此选项可以多次使用，以加载多个配置文件。
--libcurl <文件>
在任何普通 cURL 的命令行附加此选项，可将 livcurl 使用的源代码写入文件，就像您在命令行操作中做的一样！
注意：可能不适当-F 选项和发送多重 formpost ，所以在这种情况下，输出程序将丢失 curl_formadd (3) 所必要的调用，并有可能更多其它的丢失。

假如此选项已被使用过多次，则将使用最后一次给定的文件名。(在7.16.1版加入)
--limit-rate <速度>
指定 cURL 使用的最高传输率。如果您的带宽有限且您不想让您的传输占据整个带宽，那么此功能是有用的。
除非给定后缀，否则默认速度单位为字节/秒。加上' k '或' K '后缀则以千字节计算， 'm'或 M '则以兆字节计算，而'g'或' G'则以千兆字节计算。例子： 200K， 3m 和1G。
给定的速率是整体传输过程的平均值。这意味着，cURL 可能在短时间有爆发传输速度，但随着时间的推移它不会超过给定的速率。
如果您还使用-Y/--speed-limit 选项，该选项将优先考虑，并可能使限速略有下降，以帮助保持速度限制逻辑工作。
假如此选项已被使用过多次，则使用最后一次的设置。
-l/--list-only
(FTP)当列出 FTP 目录，此开关显示只有名称的列表。尤其是当普通目录视图没有使用标准外观或格式，而您需要机器解析 FTP 目录内容时，此选项非常有用。此选项发送 FTP NLST 命令。有些 FTP 服务器只对 NLST 命令列出文件列表，不包括子目录和符号连接。
--local-port <端口号>[-num]
设置连接使用的首选端口号或本地端口范围 。请注意，端口号是一种稀缺资源，繁忙时，请将端口范围缩小来避免不必要的连接失败。(在7.15.2版加入)
-L/--location
(HTTP / HTTPS)如果服务器报告所请求的网页已经转移到了其它的位置(位置的指示根据：HTTP-header 和3XX 响应代码) ，此选项将使 cURL 重新向新位置发起请求。如果与-i/--include 或-I/--head 一起使用 ，将显示所有请求的的网页 HTTP-header。当使用身份验证时，cURL 只向初始的主机发送凭据。如果需要 cURL 重定向到不同的主机，将无法拦截用户名+密码。就如何改变这种情况又见--location-trusted。您可以使用--max-redirs 选项限制后续重定向数目。
当 cURL 跟随重定向且请求不是简单的 GET (例如 POST 或 PUT) ，如果 HTTP 响应码是301、302或303的话，它将做 GET 请求。如果响应代码是任何其他3xx 代码，cURL 将重新使用相同的方法发送请求。
--location-trusted
(HTTP / HTTPS)与-L/--location 相同 ，但将允许将用户名+密码发送到该站点可能重定向到的所有主机。这可能会也可能不会引入安全漏洞，如果网站将您重定向到一个您会发送您的身份验证信息(这种情况下是明文的 HTTP 的 Basic 认证)到该位置的网站。
--max-filesize <字节数>
指定下载文件的最大尺寸(以字节为单位)。如果请求的文件尺寸大于此值，传输将无法启动，cURL 返回退出代码63。
注意：在下载期间，文件尺寸并不总是已知的。对于这样的文件，哪怕文件传输大小最终超过给定限制，此选项也不会产生作用。此选项对 FTP 和 HTTP 传输都有效。
-m/--max-time <秒>
您允许操作花费的最大时间，以秒计算。这是非常有用的，以防止您的批处理工

作由于挂了几个小时，导致网络速度慢或链接丢失。又见--connect-timeout 选项。假如此选项已被使用过多次，则使用最后一次的设置。
-M/--manual
手册。显示帮助手册。
-n/--netrc
使 cURL 在用户 HOME 目录中扫描.netrc (在 Windows 下是_netrc)文件，查找登录名和密码。这通常是用于 Unix 操作系统的 FTP。如果是 HTTP ，cURL 将使用户认证。文件格式详见 netrc(4)或 FTP(1)。如果文件权限(不应该同为 world-或 group-可读)不正确，cURL 不会给出任何反馈。环境变量“HOME”通常用来寻找主目录。
一个迅速和非常简单的例子，如何设置一个.netrc 文件 ，使 cURL 使用用户名'myself'和密码'secrct'连接到 FTP 服务器 host.domain.com 应类似于：
machine host.domain.com login myself password secret
--netrc-optional
与--netrc 非常相似，但这一选择使 .netrc 变为可选的，而不是--netrc 选项强制性的。
--negotiate
(HTTP)启用 GSS-Nagotiate 验证。GSS-Nagotiate 方法是由微软设计，并在他们的 Web 应用程序中使用。这主要是指支持 Kerberos5身份认证，但还可使用另一个验证方法。欲了解更多信息，请参阅 IETF 草案，draft-brezak-spnego-http-04.txt。如果您需要为您的代理身份验证开启 Negotiate，请使用--proxy-negotiate 选项 。此选项要求 GSSAPI 支持编译的库文件。这并不是很常见。使用-V/--version ，看看您的版本是否支持 GSS-Negotiate。
当使用此选项时，您还必须提供一个虚假的-u/--user 选项来激活身份验证代码。发送'-u :'当做用户名与密码已足够，- u 选项实际没有使用。
如果此选项被多次使用，下列事件不作任何区别。
-N/--no-buffer
禁用输出流的缓冲。在正常工作的情况下，cURL 将使用标准的缓冲输出流。将产生的效果是将数据以块的形式输出，当数据到达时是不必精确的。使用此选项将禁用该缓冲。
请注意，这是记录在案的否定选项。您可使用--buffer 选项来执行该缓冲。
--no-keepalive
禁止 TCP 连接时使用保持活动信息，默认情况下 cURL 是允许使用的。
请注意，这是记录在案的否定选项。您可使用--keepalive 选项执行保持活动。
--no-sessionid
(SSL)禁用 cURL 使用 SSL 会话标识(session_ID)缓存。默认情况下所有传输都是使用该缓存的。请注意，当重用 SSL 会话缓存时，虽然不会破坏任何传输，但是有可能会破坏 SSL 实现。可能需要禁用此选项您才可能完成该操作。(在7.16.0 版加入)
请注意，这是记录在案的否定选项。您可以因此使用--sessionid 执行会话标识缓存。
- noproxy <不需代理的主机列表>
无需使用代理的主机列表，如有指定，则使用逗号分隔。单一通配符是*号字符，将匹配所有主机，并禁用代理。此列表中的每个名字应符合一个包含主机名的域

名，或主机本身。例如，local.com 将匹配 local.com、local.com:80和 www.local.com，但不匹配 www.notlocal.com。(在7.19.4版加入) 。
--ntlm
(HTTP)启用 NTLM 验证。NTLM 是由微软设计并在其 IISIIS Web 服务器中使用的身份验证方法。这是一个私有的协议，由一帮聪明家伙做了逆向工程。cURL 上的实现基于他们的努力。这种行为实际是不值得支持的，您应当鼓励使用 NTLM 的人切换到其它大众化并有记录的身份验证方法，比如 Digest 来代替 NTLM。
如果您想启用 NTLM 做为您的代理身份验证，则使用 --proxy-ntlm 选项。
此选项要求编译时使用 SSL 库支持。使用-V/--version，看看您的 cURL 是否支持 NTLM 身份验证。
如果此选项被多次使用，下列事件不作任何区别。
-o/--output <文件>
将获取的远程文件输出到文件中，而不是输出到屏幕上(stdout)。如果您使用{}或[]定义了批量获取文件，则您可以在<文件>选项中用#后跟随数字的格式来定义文件名。这样实际文件名中的#将被多个 URL 中的当前字符串代替。如：
curl http://{one,two}.site.com -o "file_#1.txt" 则输出 file_one 和 file_two 文件
或使用多个变量如：
curl http://{site,host}.host[1-5].com -o "#1_#2" #1将被"site"或"host"代替，#2将被1-5之间的数字代替
您处理多少条 URL 就可以使用多少次该选项。
又见--create-dirs 来动态建立本地目录。指定输出到'-'(单破折号)将强制输出到屏幕上(stdout)。
-O/--remote-name
直接以远程文件名保存本地文件。(只使用在 URL 中指定了文件名部分，不包括路径。)
用于保存的远程文件由给定 URL 提取。
您处理多少条 URL 就可以使用多少次该选项。
--remote-name-all
此选项改变默认动作到所有给定的待处理 URL，就像为每条 URL 指定 -O/--remote-name 参数一样。因此，如果您要在使用了--remote-name-all 后再为某个特定 URL 禁用它时，您必须使用"-o -"或--no-remote-name 选项 。(在7.19.0 版加入)
--pass <密码>
(SSL / SSH)私钥的密码。
假如此选项已被使用过多次，则使用最后一次的设置。
--post301
当跟随301重定向时，告知 cURL 遵循 RFC 2616/10.3.2并不要将 POST 请求转换成 GET 请求。不符合 RFC 的行为是普遍存在于 Web 浏览器中的，因此 cURL 默认做上述转换来保持一致。然而，一台服务器有可能做一个 POST 请求来保持在这样一个重定向后的 POST。此选项只有在使用-L/--location 时有意义 (在7.17.1版加入)
--post302
当跟随302重定向时，告知 cURL 遵循 RFC 2616/10.3.2并不要将 POST 请求转换

成 GET 请求。不符合 RFC 的行为是普遍存在于 Web 浏览器中的，因此 cURL 默认做上述转换来保持一致。然而，一台服务器有可能做一个 POST 请求来保持在这样一个重定向后的 POST。此选项只有在使用-L/--location 时有意义 (在7.19.1版加入)
--proxy-anyauth
告知 cURL 在连接给定代理服务器时挑选一个合适的身份验证方法。可能会导致额外的请求/应答往返。(在7.13.2 版加入)
--proxy-basic
告知 cURL 在连接给定代理服务器时选择基本 HTTP 身份验证。使用--basic 选项为远程主机开启基本 HTTP 验证。基本身份验证是 cURL 使用代理服务器时的默认身份验证方法。
--proxy-digest
告知 cURL 在连接给定代理服务器时选择 Http Digest 身份验证。使用--digest 选项为远程主机开启 HTTP Digest 验证。
--proxy-negotiate
告知 cURL 在连接给定代理服务器时选择 HTTP Negotiate 身份验证。使用 --negotiate 选项为远程主机开启 HTTP Negotiate 验证。(在7.17.1版中加入)
--proxy-ntlm
告知 cURL 在连接给定代理服务器时选择 HTTP NTLM 身份验证。使用--ntlm 选项为远程主机使用 NTLM 验证。
--proxy1.0 <代理主机[:端口]>
使用指定的 HTTP 1.0代理。如未指定端口号，则假定使用1080端口。
和 HTTP 代理选项(-x/--proxy)的唯一区别，是此选项尝试指定 HTTP 1.0协议而不是默认的 HTTP 1.1协议的 CONNECT 通过代理服务器。
-p/--proxytunnel
当使用 HTTP 代理(-x/--proxy)时，此选项将试图使非 HTTP 协议通过代理隧道，而非使用类似 HTTP 的操作行为。代理隧道的方法是通过 HTTP 代理的 CONNECT 发起请求，请求代理允许直接连接到 cURL 需要通过的隧道的远程端口号来实现的。
--pubkey<公钥>
(SSH)公钥的文件名。允许您在单独的文件中提供您的公钥。
假如此选项已被使用过多次，则使用最后一次的设置。
-P/--ftp-port <地址>
(FTP)当连接到 FTP 服务器时，此选项让 cURL 使用主动模式。在实践中，cURL 使服务器连接到客户端的指定地址和端口，而被动模式要求服务器为其设置一个 IP 地址和端口来连接。 <地址>应该是以下之一：
界面
即"eth0" 指定您要使用的网络界面 ip 地址(只在 Unix 下有效)
IP 地址
即“ 192.168.10.1 ”指定确切的 IP 地址
主机名称
即“ my.host.domain ”指定主机名称
-
使 cURL 选择已被当前控制连接使用的相同 IP 地址。

假如此选项已被使用过多次，则使用最后一次的设置。--ftp-pasv 选项禁用 PORT 命令。使用参数--disable-eprt 尝试用 EPRT 命令代替 PORT 命令。EPRT 其实就是 PORT 的升级版。
从7.19.5版开始 ，你可以在地址的右侧附加 ":[开始]-[结束]"，指定 cURL 使用的 TCP 端口范围。从低到高指定端口的范围。单一端口号也能工作，但是请注意如果端口无法使用，会增加失败的风险。
-q
如果作为命令行上第一个参数，则不会读取和使用 crulrc 配置文件。配置文件的默认搜索路径详情见-K/--config，。
-Q/--quote <命令>
(FTP / SFTP)发送任意指令到远程 FTP 或 SFTP 服务器。quote 命令在传输发生之前发送(就在 FTP 传输中最初的 PWD 命令后，必须准确) 。要使命令在成功传输后发送，则为它们加前缀破折号'- '。要使命令在 libcurl 改变工作目录后而又在传输命令前发送，则为它们加前缀'+'(只支持 FTP 协议) 。您可以指定任意数量的命令。如果服务器返回一个失败的命令，整个操作将中止。您必须以 RFC959 定义的正确语法向 FTP 服务器发送 FTP 命令，或者向 SFTP 服务器发送下列命令。此选项可以多次使用。
SFTP 一个二进制协议。不同于 FTP，libcurl 解释 SFTP 的 quote 命令后再将它们发送到服务器。以下列出所有支持的 SFTP quote 命令：
chgrp group file
该 chgrp 命令将以 group 操作符指定的组 ID 设置给以 file 操作符指定的文件名。group 操作符应为一个十进制整数的组 ID。
chmod mode file
该 chmod 命令修改文件属性。mod 操作符应为一个八进制整数的属性数字。
chown user file
该 chown 命令设置以 file 操作符定义的文件所有者为以 user 操作符指定的用户 ID。user 操作符应为一个十进制整数的用户 ID。
ln source_file target_file
ln 和 symlink 命令创建一个符号链接，从 target_file 位置指向 source_file 位置。
mkdir directory_name
mkdir 命令创建名为 directory_name 的目录。
pwd
pwd 命令返回当前工作目录的绝对路径。
rename source target
rename 命令将源(source)目录或文件的名称重命名为目标(target)目录或文件的名称。
rm file
使用 rm 命令删除 file 操作符指定的文件。
rmdir directory
rmdir 命令移除 directory 操作符指定的非空目录。
symlink source_file target_file
见 ln。
--random-file <文件>
(SSL)指定包含将被视为随机数据的文件路径。这些数据将作为 SSL 连接的随机

数引擎的种子。又见--egd-file 选项。
-r/--range <范围>
(HTTP/FTP/SFTP/FILE)该选项指定从 HTTP/1.1、FTP 或 SFTP 服务器又或本地文件下载字节的范围，常应用于分块下载文件。指定范围可有多种表示。
0-499指定前500字节
500-999指定第二500字节
-500指定最后500字节
9500 -指定从9500字节开始的全部字节。
0-0,-1只指定第一个和最后一个字节(*) （H）
500-700,600-799从偏移500字节开始的300字节(H)
100-199,500-599指定两个不同的100字节的范围(*)（H)
(*) =请注意，这将导致服务器回复多重响应！
在‘开始-停止’这样的语法中只能使用数字(0-9)。如果指定了非数字的字符，则取决于服务器设置，会有不可预料的回应。
您还应该了解，许多 HTTP/1.1服务器没有启用此项功能，因此，当您试图获得该范围的文件内容时，会得到整个文件。
FTP 和 SFTP 分块下载只支持简单的‘开始-停止’语法(其中一个数字是可以省略的)。FTP 的使用取决于扩展 FTP 命令 SIZE。
假如此选项已被使用过多次，则使用最后一次的设置。
--raw
使用此选项禁用所有内部内容的 HTTP 解码或传输编码，并转而让其以原格式传递。(在7.16.2版加入)
-R/--remote-time
使用此选项，libcurl 将试图寻找远程文件的时间戳，使本地文件获得与其相同的时间戳。
--retry <次数>
设置 cURL 在传输时遇到暂态错误后重试的次数，超过此次数则放弃传输。将次数设置为0则使 cURL 不做重试(这是默认值)。暂态误差是指：传输超时、FTP 5xx 响应码或 HTTP 5xx 响应码。
当 cURL 重试传输时，先等待一秒。然后对于所有即将发起的重试请求，此等待时间将加倍直到到达10分钟。此10分钟的等待将会是剩余重试请求之间的延迟。使用--retry-delay 选项禁用此退避时间(Backoff Time)指数算法。要限制允许重试的总时间，又见--retry-max-time。(在7.12.3版加入)
如果多次使用此选项，则最后一次的设定的起作用。
--retry-delay <秒>
此选项设置传输过程遇到暂态错误时，cURL 在每次重试前等待的时间(它改变了重试请求之间默认的退避时间(Backoff Time)算法)。只有当使用--retry 选项时，此选项才起作用。将此设置为0则 cURL 使用默认的退避时间(Backoff Time)。(在7.12.3版加入)
如果多次使用此选项，则最后一次的设定起作用。
--retry-max-time <秒>
在发起第一次传输前，重试计时器将重置。重试将像往常一样进行(见--retry)，只要计时器还没有到达此给定限制。请注意，如果计时器还没有达到限制时间，该请求将被执行。在执行期间，它需要的时间可能比给定的更长。要限制单次请

求的最大时间，请使用-m/--max-time 参数 。将此选项设置为0则不作超时重试。(在7.12.3版加入)
如果多次使用此选项，则最后一次的设定起作用。
-s/--silent
安静模式。不显示进度表或错误信息。使 cURL 不反馈信息。
-S/--show-error
当与-s 选项一起使用时，cURL 在其出错时显示一条出错信息。
--socks4 <主机[:端口]>
使用指定的 SOCKS4代理。如未指定端口号，则假定是1080端口。(在7.15.2版加入)
此选项会覆盖任何先前使用的-x/--proxy 选项 ，因为它们是相互排斥的。
假如此选项已被使用过多次，则使用最后一次的设置。
--socks4a <主机[:端口]>
使用指定的 SOCKS4a 代理。如未指定端口号，则假定是1080端口。(在7.18.0 版加入)
此选项会覆盖任何先前使用的-x/--proxy 选项 ，因为它们是相互排斥的。
假如此选项已被使用过多次，则使用最后一次的设置。
--socks5-hostname <主机[:端口]>
使用指定的 SOCKS5代理(并让代理解析主机名称) 。如未指定端口号，则假定是1080端口。(在7.18.0 版加入)
此选项会覆盖任何先前使用的-x/--proxy 选项 ，因为它们是相互排斥的。
假如此选项已被使用过多次，则使用最后一次的设置。(此选项曾错误地记录和用作--socks 没有附加的数量。)
--socks5 <主机[:端口]>
使用指定的 SOCKS5代理——但在本地解析主机名。如未指定端口号，则假定是1080端口。
此选项会覆盖任何先前使用的-x/--proxy 选项 ，因为它们是相互排斥的。
假如此选项已被使用过多次，则使用最后一次的设置。(此选项曾错误地记录和用作--socks 没有附加的数量。)
--socks5-gssapi-service <服务名>
socks 服务器的默认服务名是 rcmd/server-fqdn。此选项允许您改变它。
范 例 ： --socks5 proxy-name --socks5-gssapi-service sockd would use sockd/proxy-name --socks5 proxy-name --socks5-gssapi-service sockd/real-name would use sockd/real-name for cases where the proxy-name does not match the princpal name。(在7.19.4版加入) 。
--socks5-gssapi-nec
作为 gssapi negotiation 的一部分，协商保护模式。RFC1961的第4.3/4.4小节中提到此协议应是受保护的(protected)，但 NEC 参照实现中则不这么做。--socks5-gssapi-nec 选项允许协商保护模式的未受保护的交换。(在7.19.4 版加入)
--stderr <文件>
将输出到标准错误(stderr)的信息写入指定文件。如果文件名指定为纯文本的'-'，则写入到标准输出(stdout)。此选项在您使用 decent redirecting capabilities 的 shell 时是无用的。
假如此选项已被使用过多次，则使用最后一次的设置。

--tcp-nodelay
打开 TCP_NODELAY 选项。详细了解此选项，见 curl_easy_setopt(3)手册页。(在 7.11.2 版加入)
-t/--telnet-option <OPT=val>
为 telnet 协议传递选项。支持的选项有：
TTYPE=<term>设置终端类型。
XDISPLOC=<X display>设置 X 显示位置。
NEW_ENV=<var,val>设置一个环境变量。
-T/--upload-file <文件>
传输指定的本地文件到远程的 URL。如果 URL 中不指定文件部分，则 cURL 将使用本地文件名。请注意，您必须在最后一个目录后附加一个/，以说明确实没有提供文件名，否则 cURL 会把最后一级目录名当做文件名称。这是传输操作失败最可能的原因。如果是在 HTTP(S)中，则将使用 PUT 命令。
如果<文件>为"-"(单破折号)，则使用标准输入（stdin）来代替给定的文件。
您可在命令行上为每个 URL 指定一个-T 选项。每一个-T + URL 指定上传什么，上传到哪里。cURL 还支持- T 参数的“通配符”，也就是说，您能用过在使用 URL 格式中支持的通配符类型来上传多个文件到一个 URL，例如：
curl -T "{file1,file2}" http://www.uploadtothissite.com
甚至
curl -T "img[1-1000].png" ftp://ftp.picturemania.com/upload/
--trace <文件>
将所有传入和传出的数据，包括描述性信息的完整跟踪转储到给定的输出文件中。使用"-"作为文件名则将输出发送到标准输出(stdout)。
此选项会覆盖上一个-v/--verbose 或 --trace-ascii 选项的设置。
假如此选项已被使用过多次，则使用最后一次的设置。
--trace-ascii <文件>
将所有传入和传出的数据，包括描述性信息的完整跟踪转储到给定的输出文件中。使用"-"作为文件名则将输出发送到标准输出(stdout)。
与--trace 非常相似，但没有输出十六进制部分，而只在转储中显示了 ASCII 部分。这种体积更小的转储文件使未经训练的人士更易阅读。
此选项会覆盖上一个-v/--verbose 或 --trace 选项的设置。
假如此选项已被使用过多次，则使用最后一次的设置。
--trace-time
在 cURL 显示的每一条 trace 或 verbose 行的开始处添加一个时间标记。(在7.14.0 版加入)
-u/--user <用户名:密码>
为服务器身份验证指定用户名和密码。覆盖-n/--netrc 和--netrc-optional 选项 。
如果你只给定用户名(不输冒号)，cURL 会提示输入密码。
如果您使用了一个开启 SSPI 的 cURL 执行文件并做 NTLM 身份验证，您可强制 cURL 从您当前环境下获取用户名与密码，方法是在此选项后加一个冒号："-u :" 。
假如此选项已被使用过多次，则使用最后一次的设置。
-U/--proxy-user <用户名:密码>
为代理服务器身份验证指定用户名和密码。

如果您使用了一个开启 SSPI 的 cURL 执行文件并做 NTLM 身份验证，您可强制 cURL 从您当前环境下获取用户名与密码，方法是在此选项后加一个冒号："-u :" 。
假如此选项已被使用过多次，则使用最后一次的设置。
--url <URL>
指定要获取的 URL 地址。此选项主要是方便您在配置文件中指定 URL 时使用。此选项可使用任意多次。使用-o/--output 或-O/--remote-name 选项来控制此 URL 写往何处。
-v/--verbose
时获取过程显示更多的信息。对调试工作很有用。以'>'开头的行是指 cURL 发出的“头数据(header data)”，以 '<'开头的行是指 cURL 收到的“头数据(header data)”。这些在一般情况下是不可见的。以 '*' 开头的行是指 cURL 提供的额外信息。
请注意，如果您只想输出 HTTP header，那么-i/--include 可能是您需要的选项。
如果你觉得此办法还不能给您足够的细节，可以考虑使用--trace 或--trace-ascii 来代替。
此选项会覆盖之前使用的--trace-ascii--trace-ascii 或--trace 选项。
-V/--version
显示关于 cURL 的信息以及它使用的 libcurl 版本。
第一行包含 cURL 的完整版本号、可执行文件链接的 libcurl 和其他第三方库。
第二行(以"Protocols:"开始)显示所有 libcurl 报告支持的协议。
第三行(以"Features:"开始)显示 libcurl 报告提供的具体特性。提供的功能包括：
IPv6
您可使用 IPv6。
krb4
支持 FTP 的 Krb4。
SSL
支持 HTTPS 和 FTPS。
libz
支持通过 HTTP 自动解压被压缩的文件。
NTLM
支持 NTLM 身份验证。
GSS-Negotiate
支持 FTP 的 Negotiate 身份验证和 krb5。
Debug
cURL 使用除错选项的 libcurl 编译。允许更多的错误跟踪和内存调试信息，cURL 开发专用!
AsynchDNS
使 cURL 使用异步名称解析。
SPNEGO
支持 SPNEGO 协商身份验证。
Largefile
cURL 支持大于2GB 的文件传输 。
IDN
cURL 支持 IDN 国际域名。

SSPI
支持 SSPI。如果您使用 NTLM 并设置了一个空用户名，cURL 将使用您当前的用户名和密码做身份验证。
-w/--write-out <格式>
定义当成功传输完成后显示在标准输出(stdout)的信息。该信息的格式可能是混合了纯文本与任意数目变量的字符串。该字符串可被指定为"string"，其中"@filename"表示输出 filename 文件中的数据，"@-"表示输出用户写入标准输入(stdin)的数据。
在输出格式中呈现的变量会被 cURL 用其认为合适的值或文本取代，分述如下。所有变量都以%{变量名}这样的格式定义，用%%来输出正常的%。用\n 表示换行、\r 表示回车、\t 表示一个制表符宽度(tab)
注意： %符号在 Win32的环境下是一个特殊的符号，使用此选项时，所有出现%的地方必须以%%代替。
常用变量名有：
url_effective 最后获取的 URL。此信息当您告知 cURL 来跟随 location: headers 时是最有用的。
http_code 上一次 HTTP(S)或 FTP(S)操作返回的响应码。在 7.18.2 版加入的 response_code 显示同样的信息。
http_connect 在最后一次对 cURL 的 CONNECT 请求的响应(从代理)中发现的数值代码。 （在7.12.4版加入)
time_total 全部操作耗费的时间，单位为秒。精确到毫秒。
time_namelookup 从开始到域名解析完成耗费的时间，单位为秒。
time_connectTCP 连接远程主机(或代理服务器)所耗时间，单位为秒。
time_appconnectSSL/SSH/等与远程主机连接/握手完成花费的时间，单位为秒。(在7.19.0 版加入)
time_pretransfer 从开始到文件将要传输前花费的时间，单位为秒。包括指定的协议所有预传输命令和 negotiations。
time_redirect 所有重定向步骤的时间，包域名解析、连接、预传输和最后事务开始前的传输，单位为秒。time_redirect 显示多重重定向的完整执行时间。(在7.12.3 版加入)
time_starttransfer 从开始到第一个字节将被传输前耗费的时间，单位为秒。这包括 time_pretransfer 和服务器需要的运算结果的时间。
size_download 下载的总字节数。
size_upload 上传的总字节数。
size_header 下载的 header 的总字节数。
size_request 发送的 HTTP 请求的总字节数。
speed_downloadcurl 成功下载的平均下载速度。
speed_uploadcurl 成功上传的平均上传速度。
content_type 如果有，显示请求文档的 Content-Type
num_connects 最近一次传输中的连接数目。(在7.12.3版加入)
num_redirects 跟随请求的重定向数目。 (在7.12.3版加入)
redirect_url 当未使用-L 选项发起跟随重定向的 HTTP 请求时，此变量将显示重定向到的实际 URL。(在7.18.2版加入)
ftp_entry_path 初始路径 libcURL 结束时，在登录到远程 FTP 服务器。(在7.15.4

版加入)
ssl_verify_result 请求的 SSL peer 证书校验结果。0表示校验成功。(在7.19.0 版加入)
假如此选项已被使用过多次，则使用最后一次的设置。
-x/--proxy <代理主机[:端口]>
使用指定的 HTTP 代理。如未指定端口号，则假定是1080端口。
此选项会覆盖目前环境变量中的代理设置。如果有环境变量设置了代理，可以设置代理为""来覆盖此环境变量中的代理设置。
请注意，所有通过 HTTP 代理的操作都会透明的转化为 HTTP 协议。这意味着某些特定协议的操作将会无效。这是没问题的，如果使用-p/--proxytunnel 选项设定通过隧道代理。
从7.14.1版开始，代理主机可被指定与代理环境变量，包括协议前缀(http://)和嵌入的用户名与密码完全一样的方式。
假如此选项已被使用过多次，则使用最后一次的设置。
-X/--request <命令>
(HTTP)在与 HTTP 服务器连接时使用一个自定义的请求方法。指定的请求将被用来代替使用的其他方法(默认是 GET 方法)。详细资料和解释请阅读 HTTP 1.1 规范。
(FTP)在从 FTP 服务器获取文件列表时，使用一个自定义的 FTP 命令代替 LIST 命令。
假如此选项已被使用过多次，则使用最后一次的设置。
-y/--speed-time <时间>
如果在<时间>设定的范围内，下载速度慢于指定的字节每秒的速度则中止下载。如果使用此选项，除非同时用-Y 选项设置，否则默认的传输速度为1(字节/秒)。此选项控制传输，使其不会影响缓慢的连接等。如果您关心的是这个，请尝试--connect-timeout 选项。
假如此选项已被使用过多次，则使用最后一次的设置。
-Y/--speed-limit <速度>
如果下载速度在 speed-time 设置的秒数里慢于此选项指定的速度(字节/秒)，则下载中止。speed-time 使用-y 选项设置，默认为30(秒)。
假如此选项已被使用过多次，则使用最后一次的设置。
-z/--time-cond <日期表达式>
(HTTP/FTP)请求最后修改时间迟于或早于给定日期的文件。日期表达式可以是所有类型的日期字符串。如果指定的日期表达式与已知的任意表达式都不相符，则会尝试从给定的文件名中来获取。日期表达式详情见 curl_getdate(3)手册页。
使用破折号(-)开始的日期表达式则请求此日期之前的文档，默认是获取比指定日期/时间更新的文档。
假如此选项已被使用过多次，则使用最后一次的设置。
--max-redirs <数量>
设定允许的重定向的最大数量。如果使用-L/--location 选项，则此选项可以用来使 cURL 避免“超越边界的”重定向。默认情况下，限制设置为50个重定向。将此选项设置为-1则是无限。
假如此选项已被使用过多次，则使用最后一次的设置。
-0/--http1.0

(HTTP)强制 cURL 使用 HTTP 1.0而不是其默认的 HTTP 1.1来发出请求 。
-1/--tlsv1
(SSL)强制 cURL 使用 TLS 第1版协议与远程的 TLS 服务器进行协商。
-2/--sslv2
(SSL)强制 cURL 使用 SSL 第2版协议与远程的 SSL 服务器进行协商。
-3/--sslv3
(SSL)强制 cURL 使用 SSL 第3版协议与远程的 SSL 服务器进行协商。
-4/--ipv4
如果 libcurl 是能够解析多 IP 地址的版本(如果它是 IPv6兼容的)，此选项告知 libcurl 只解析指向 IPv4的地址。
-6/--ipv6
如果 libcurl 是能够解析多 IP 地址的版本(如果它是 IPv6兼容的)，此选项告知 libcurl 只解析指向 IPv6的地址。
-#/--progress-bar
使 cURL 用一个进度条代替默认的统计数据来显示进度信息。

文件

~/.curlrc
默认的配置文件，详情见-K/--config。

环境

环境变量可以使用小写或大写。小写版本的优先。 http_proxy 是一个例外，因为它仅适用于小写。
http_proxy [协议://]<主机>[:端口]
为 HTTP 设置代理服务器 。
HTTPS_PROXY [协议://]<主机>[:端口]
为 HTTPS 设置代理服务器 。
FTP_PROXY [协议://]<主机>[:端口]
为 FTP 设置代理服务器 。
ALL_PROXY [协议://]<主机>[:端口]
为无指定协议的代理设定代理服务器。
NO_PROXY <逗号分隔的主机列表>
不使用任何代理的主机列表。如果只设置为星号' * '，它匹配所有主机。

退出码

这里有许多可能会出现在恶劣条件下的不同错误代码和相应信息。在本文写作之时，退出代码有：
1
未支持的协议。此版 cURL 不支持这一协议。
2
初始化失败。

3
URL 格式错误。语法不正确。
5
无法解析代理。无法解析给定代理主机。
6
无法解析主机。无法解析给定的远程主机。
7
无法连接到主机。
8
FTP 非正常的服务器应答。cURL 无法解析服务器发送的数据。
9
FTP 访问被拒绝。服务器拒绝登入或无法获取您想要的特定资源或目录。最有可能的是您试图进入一个在此服务器上不存在的目录。
11
FTP 非正常的 PASS 回复。cURL 无法解析发送到 PASS 请求的应答。
13
FTP 非正常的的 PASV 应答，cURL 无法解析发送到 PASV 请求的应答。
14
FTP 非正常的227格式。cURL 无法解析服务器发送的227行。
15
FTP 无法连接到主机。无法解析在227行中获取的主机 IP。
17
FTP 无法设定为二进制传输。无法改变传输方式到二进制。
18
部分文件。只有部分文件被传输。
19
FTP 不能下载/访问给定的文件， RETR (或类似)命令失败。
21
FTP quote 错误。quote 命令从服务器返回错误。
22
HTTP 找不到网页。找不到所请求的 URL 或返回另一个 HTTP 400或以上错误。此返回代码只出现在使用了-f/--fail 选项以后。
23
写入错误。cURL 无法向本地文件系统或类似目的写入数据。
25
FTP 无法 STOR 文件。服务器拒绝了用于 FTP 上传的 STOR 操作。
26
读错误。各类读取问题。
27
内存不足。内存分配请求失败。
28
操作超时。到达指定的超时期限条件。
30
FTP PORT 失败。PORT 命令失败。并非所有的 FTP 服务器支持 PORT 命令，请

尝试使用被动(PASV)传输代替！
31
FTP 无法使用 REST 命令。REST 命令失败。此命令用来恢复的 FTP 传输。
33
HTTP range 错误。range "命令"不起作用。
34
HTTP POST 错误。内部 POST 请求产生错误。
35
SSL 连接错误。SSL 握手失败。
36
FTP 续传损坏。不能继续早些时候被中止的下载。
37
文件无法读取。无法打开文件。权限问题？
38
LDAP 无法绑定。LDAP 绑定(bind)操作失败。
39
LDAP 搜索失败。
41
功能无法找到。无法找到必要的 LDAP 功能。
42
由回调终止。应用程序告知 cURL 终止运作。
43
内部错误。由一个不正确参数调用了功能。
45
接口错误。指定的外发接口无法使用。
47
过多的重定向。cURL 达到了跟随重定向设定的最大限额跟
48
指定了未知 TELNET 选项。
49
不合式的 telnet 选项。
51
peer 的 SSL 证书或 SSH 的 MD5指纹没有确定。
52
服务器无任何应答，该情况在此处被认为是一个错误。
53
找不到 SSL 加密引擎。
54
无法将 SSL 加密引擎设置为默认。
55
发送网络数据失败。
56
在接收网络数据时失败。
58

本地证书有问题。
59
无法使用指定的 SSL 密码。
60
peer 证书无法被已知的 CA 证书验证。
61
无法辨识的传输编码。
62
无效的 LDAP URL。
63
超过最大文件尺寸。
64
要求的 FTP 的 SSL 水平失败。
65
发送此数据需要的回卷(rewind)失败。
66
初始化 SSL 引擎失败。
67
用户名、密码或类似的信息未被接受，cURL 登录失败。
68
在 TFTP 服务器上找不到文件。
69
TFTP 服务器权限有问题。
70
TFTP 服务器磁盘空间不足。
71
非法的 TFTP 操作。
72
未知 TFTP 传输编号(ID)。
73
文件已存在(TFTP) 。
74
无此用户(TFTP) 。
75
字符转换失败。
76
需要字符转换功能。
77
读 SSL 证书出现问题(路径？访问权限？) 。
78
URL 中引用的资源不存在。
79
SSH 会话期间发生一个未知错误。
80

未能关闭 SSL 连接。
82
无法加载 CRL 文件，丢失或格式不正确(在7.19.0版中增加) 。
83
签发检查失败(在7.19.0版中增加) 。
XX
更多信息错误代码会在未来版本中出现在这里。现有的条目将永不改变。

作者/贡献者

主要作者为丹尼尔斯腾伯格，但完整的贡献者名单在单独的 THANKS 文件中。

万维网

http://cURL.haxx.se

FTP

ftp://ftp.sunet.se/pub/www/utilities/cURL/

也可参见

FTP (1) wget (1)
网页更新于09年8月7日
translated by MagicNight(magicnight#gmail.com)
blog.magicnight.cn