Похожие:
Справочная страница
ЧАСТО задаваемые вопросы

учебное пособие по curl

Простое использование

Получить главную страницу с веб-сервера:

```
curl https://www.example.com/
```

Получить файл README с FTP-сервера:

```
curl ftp://ftp.funet.fi/README
```

Получить веб-страницу с сервера, используя порт 8000:

```
curl http://www.weirdserver.com:8000/
```

Получить список каталогов FTP-сайта:

```
curl ftp://ftp.funet.fi
```

Get the all terms matching curl from a dictionary:

```
curl dict://dict.org/m:curl
```

Получите определение curl из словаря:

```
curl dict://dict.org/d:curl
```

Извлекать два документа одновременно:

```
curl ftp://ftp.funet.fi/ http://www.weirdserver.com:8000/
```

Получить файл с сервера FTPS:

```
curl ftps://files.are.secure.com/secrets.txt
```

или используйте более подходящий способ FTPS для получения того же файла:

```
curl --ftp-ssl ftp://files.are.secure.com/secrets.txt
```

Получить файл с SSH-сервера с помощью SFTP:

```
curl -u username sftp://example.com/etc/issue
```

Получите файл с SSH-сервера с помощью SCP, используя закрытый ключ (не защищенный паролем) для аутентификации:

```
curl -u username: --key ~/.ssh/id_rsa scp://example.com/~/file.txt
```

Получите файл с SSH-сервера с помощью SCP, используя закрытый ключ (защищенный паролем) для аутентификации:

```
curl -u username: --key ~/.ssh/id_rsa --pass private_key_password
scp://example.com/~/file.txt
```

Get the main page from an IPv6 web server:

```
curl "http://[2001:1890:1112:1::20]/"
```

Получить файл с SMB-сервера:

```
curl -u "domain\username:passwd" smb://server.example.com/share/file.txt
```

Загрузить в файл

Получите веб-страницу и сохраните в локальном файле с определенным именем:

```
curl -o thatpage.html http://www.example.com/
```

Получите веб-страницу и сохраните в локальном файле, сделайте так, чтобы локальный файл получал имя удаленного документа (если в URL-адресе не указана часть имени файла, это приведет к сбою):

```
curl -O http://www.example.com/index.html
```

Извлеките два файла и сохраните их с их удаленными именами:

```
curl -O www.haxx.se/index.html -O curl.se/download.html
```

Использование паролей

FTP

Для файлов ftp, использующих имя и пароль, включите их в URL, например:

```
curl ftp://name:passwd@machine.domain:port/full/path/to/file
```

или укажите их с помощью -u флага, подобного

```
curl -u name:passwd ftp://machine.domain:port/full/path/to/file
```

FTPS

Это так же, как для FTP, но вы также можете указать и использовать параметры, специфичные для SSL, Для сертификатов и т.д.

Обратите внимание, что использование FTPS:// в качестве префикса является *неявным* способом, как описано в стандартах, в то время как рекомендуемый *явный* способ выполняется с помощью FTP:// и --ssl-reqd опции.

SFTP / SCP

Это похоже на FTP, но вы можете использовать --key опцию, чтобы указать закрытый ключ для использования вместо пароля. Обратите внимание, что закрытый ключ сам по себе может быть защищен паролем, который не связан с паролем для входа в удаленную систему; этот пароль указывается с помощью --pass опции. Обычно curl автоматически извлекает открытый ключ из файла закрытого ключа, но в случаях, когда curl не имеет надлежащей библиотечной поддержки, соответствующий файл открытого ключа должен быть указан с помощью --pubkey опции.

HTTP

Curl также поддерживает пользователя и пароль в URL-адресах HTTP, таким образом, вы можете выбрать файл, подобный:

```
curl http://name:passwd@machine.domain/full/path/to/file
```

или укажите пользователя и пароль отдельно, как в

```
curl -u name:passwd http://machine.domain/full/path/to/file
```

HTTP предлагает множество различных методов аутентификации, и curl поддерживает несколько из них: Basic, Digest, NTLM и Negotiate (SPNEGO). Не указывая, какой метод использовать, curl по умолчанию использует Basic. Вы также можете попросить curl выбрать наиболее безопасные из тех, которые сервер принимает для данного URL, используя --anyauth.

Обратитевнимание! Согласно спецификации URL, HTTP URL-адреса не могут содержать пользователя и пароль, поэтому этот стиль не будет работать при использовании curl через прокси, даже если curl разрешает это в другое время. При использовании прокси-сервера вы *должны* использовать -u стиль для пользователя и пароля.

HTTPS

Вероятно, чаще всего используется с частными сертификатами, как описано ниже.

Прокси

curl поддерживает как HTTP, так и SOCKS прокси-серверы с дополнительной аутентификацией. У него нет специальной поддержки для FTP-прокси-серверов, поскольку для них нет стандартов, но его все равно можно заставить работать со многими из них. Вы также можете использовать прокси-серверы HTTP и SOCKS для передачи файлов на FTP-серверы и с них.

Получите ftp-файл с помощью HTTP-прокси с именем my-proxy, который использует порт 888:

```
curl -x my-proxy:888 ftp://ftp.leachsite.com/README
```

Получите файл с HTTP-сервера, для которого требуются пользователь и пароль, используя тот же прокси, что и выше:

```
curl -u user:passwd -x my-proxy:888 http://www.get.this/
```

Некоторые прокси требуют специальной аутентификации. Укажите с помощью -U, как указано выше:

```
curl -U user:passwd -x my-proxy:888 http://www.get.this/
```

Разделенный запятыми список хостов и доменов, которые не используют прокси, может быть указан как:

```
curl --noproxy localhost,get.this -x my-proxy:888 http://www.get.this/
```

Если прокси-сервер указан с помощью --proxy1.0 вместо --proxy -x или CONNECT, то curl будет использовать HTTP / 1.0 вместо HTTP / 1.1 для любых, каких бы то ни было,, попыток.

curl также поддерживает прокси SOCKS4 и SOCKS5 с помощью --socks4 и --socks5.

Смотрите также переменные среды, поддерживаемые Curl, которые предлагают дополнительный контроль прокси.

Большинство прокси-серверов FTP настроены так, чтобы с точки зрения клиента выглядеть как обычный FTP-сервер, со специальными командами для выбора удаленного FTP-сервера. curl поддерживает -u, -Q и --ftp-account опции, которые можно использовать для настройки передач через множество FTP-прокси. Например, файл можно загрузить на удаленный FTP-сервер, используя FTP-прокси Blue Coat с параметрами:

```
curl -u "username@ftp.server Proxy-Username:Remote-Pass"
  --ftp-account Proxy-Password --upload-file local-file
  ftp://my-ftp.proxy.server:21/remote/upload/path/
```

Смотрите руководство для вашего FTP-прокси, чтобы определить форму, в которой он ожидает настройки передач, и -v опцию curl, чтобы точно видеть, что отправляет curl.

Обвязка

Получите ключевой файл и добавьте его с помощью apt-key (когда в системе, которая использует apt для управления пакетами):

```
curl -L https://apt.llvm.org/llvm-snapshot.gpg.key | sudo apt-key add -
```

'|' передает выходные данные в стандартный интерфейс. - сообщает apt-key, что файл ключа должен быть прочитан из STDIN.

Диапазоны

В HTTP 1.1 введены диапазоны байтов. Используя это, клиент может запросить получение только одной или нескольких частей указанного документа. Curl поддерживает это с помощью -r флага.

Получаем первые 100 байт документа:

```
curl -r 0-99 http://www.get.this/
```

Получаем последние 500 байт документа:

```
curl -r -500 http://www.get.this/
```

Curl также поддерживает простые диапазоны для файлов FTP. Затем вы можете указать только начальную и конечную позиции.

Получите первые 100 байт документа с помощью FTP:

```
curl -r 0-99 ftp://www.get.this/README
```

Загрузка

FTP / FTPS / SFTP / SCP

Загрузите все данные из стандартного интерфейса на указанный сервер:

```
curl -T - ftp://ftp.upload.com/myfile
```

Загрузите данные из указанного файла, войдите под именем пользователя и паролем:

```
curl -T uploadfile -u user:passwd ftp://ftp.upload.com/myfile
```

Загрузите локальный файл на удаленный сайт и используйте имя локального файла также на удаленном сайте:

```
curl -T uploadfile -u user:passwd ftp://ftp.upload.com/
```

Загрузите локальный файл, который будет добавлен к удаленному файлу:

```
curl -T localfile -a ftp://ftp.upload.com/remotefile
```

Curl также поддерживает загрузку по ftp через прокси, но только если прокси настроен на разрешение такого туннелирования. Если это произойдет, вы можете запустить curl способом, аналогичным:

```
curl --proxytunnel -x proxy:port -T localfile ftp.upload.com
```

SMB / SMBS

```
curl -T file.txt -u "domain\username:passwd"
  smb://server.example.com/share/
```

HTTP

Загрузите все данные из стандартного интерфейса на указанный HTTP-сайт:

```
curl -T - http://www.upload.com/myfile
```

Обратите внимание, что HTTP-сервер должен быть настроен на прием PUT, прежде чем это можно будет выполнить успешно.

О других способах загрузки данных по HTTP смотрите раздел публикации ниже.

Подробный / отладочный

Если curl терпит неудачу там, где этого не должно быть, если серверы не пропускают вас, если вы не можете понять ответы: используйте -v флаг, чтобы получить подробную выборку. Curl выведет много информации о том, что он отправляет и получает, чтобы пользователь мог видеть все взаимодействия клиент-сервер (но он не покажет вам фактические данные).

```
curl -v ftp://ftp.upload.com/
```

Чтобы получить еще больше подробностей и информации о том, что делает curl, попробуйте использовать опции `--trace` или `--trace-ascii` с заданным именем файла для входа, например:

```
curl --trace trace.txt www.haxx.se
```

Подробная информация

Различные протоколы предоставляют разные способы получения подробной информации о конкретных файлах / документах. Чтобы заставить curl отображать подробную информацию об одном файле, вам следует использовать опцию `-I/--head` . Он отображает всю доступную информацию в одном файле для HTTP и FTP. Информация о HTTP намного более обширна.

Для HTTP вы можете получить информацию о заголовке (ту же, что `-I` будет отображаться) перед данными, используя `-i/--include` . Curl использует опцию `-D/--dump-header` при получении файлов как с FTP, так и с HTTP, и затем сохранит заголовки в указанном файле.

Сохраните заголовки HTTP в отдельном файле (headers.txt в примере):

```
curl --dump-header headers.txt curl.se
```

Обратите внимание, что заголовки, сохраненные в отдельном файле, могут пригодиться позже, если вы хотите, чтобы curl использовал файлы cookie, отправляемые сервером. Подробнее об этом в разделе cookies.

ОПУБЛИКОВАТЬ (HTTP)

С помощью curl легко размещать данные. Это делается с помощью `-d <data>` опции. Данные post должны быть urlencoded.

Опубликуйте простую name и phone гостевую книгу.

```
curl -d "name=Rafael%20Sagula&phone=3320780" http://www.where.com/guest.cgi
```

Или автоматически URL кодирует данные.

```
curl --data-urlencode "name=Rafael Sagula&phone=3320780" http://www.where.com/guest
```

Как опубликовать форму с помощью curl, урок # 1:

Выделите все `<input>` теги в форме, которую вы хотите заполнить.

Если есть обычный пост, вы используете `-d` для публикации. `-d` принимает полную строку post, которая находится в формате

```
<variable1>=<data1>&<variable2>=<data2>&...
```

Имена переменных - это имена, заданные с помощью "name=" в `<input>` тегах, а данные - это содержимое, которое вы хотите заполнить для входных данных. Данные *должны* быть правильно

закодированы по URL. Это означает, что вы заменяете пробел на + и заменяете странные буквы на, %XX где XX - шестнадцатеричное представление ASCII-кода буквы.

Пример:

(страница расположена по адресу http://www.formpost.com/getthis/)

```
<form action="post.cgi" method="post">
<input name=user size=10>
<input name=pass type=password size=10>
<input name=id type=hidden value="blablabla">
<input name=ding value="submit">
</form>
```

Мы хотим ввести пользователя foobar с паролем 12345.

Чтобы опубликовать это, вы вводите командную строку curl, например:

```
curl -d "user=foobar&pass=12345&id=blablabla&ding=submit"
  http://www.formpost.com/getthis/post.cgi
```

-dB то время как curl использует mime-тип application / x-www-form-urlencoded, обычно понятный CGI и аналогичный, curl также поддерживает более эффективный тип multipart / form-data. Этот последний тип поддерживает такие вещи, как загрузка файлов.

-F принимает параметры, подобные -F "name=contents". Если вы хотите, чтобы содержимое считывалось из файла, используйте @filename в качестве содержимого. При указании файла вы также можете указать тип содержимого файла, добавив ;type=<mime type> к имени файла. Вы также можете опубликовать содержимое нескольких файлов в одном поле. Например, имя поля coolfiles используется для отправки трех файлов с разными типами содержимого, используя следующий синтаксис:

```
curl -F "coolfiles=@fil1.gif;type=image/gif,fil2.txt,fil3.html"
  http://www.post.com/postit.cgi
```

Если тип содержимого не указан, curl попытается угадать расширение файла (он знает только несколько) или использовать ранее указанный тип (из более раннего файла, если в списке указано несколько файлов), или же он будет использовать тип по умолчанию application/octet-stream.

Эмулируйте форму для заполнения с помощью -F. Допустим, вы заполнили три поля в форме. Одно поле - это имя файла, который нужно опубликовать, одно поле - это ваше имя и одно поле - описание файла. Мы хотим опубликовать файл, который мы написали с именем cooltext.txt. Чтобы разрешить curl размещать эти данные вместо вашего любимого браузера, вы должны прочитать исходный HTML-код страницы формы и найти названия полей ввода. В нашем примере именами полей ввода являются file, yourname и filedescription.

```
curl -F "file=@cooltext.txt" -F "yourname=Daniel"
  -F "filedescription=Cool text file with cool text inside"
  http://www.post.com/postit.cgi
```

Чтобы отправить два файла в одном сообщении, вы можете сделать это двумя способами:

Отправьте несколько файлов в одном поле с одним именем поля:

```
curl -F "pictures=@dog.gif,cat.gif" $URL
```

Отправьте два поля с двумя именами полей

```
curl -F "docpicture=@dog.gif" -F "catpicture=@cat.gif" $URL
```

Чтобы передать значение поля буквально, не интерпретируя начальное @ или < или встроенное ;type=, используйте --form-string вместо -F . Это рекомендуется, когда значение получено от пользователя или из какого-либо другого непредсказуемого источника. В этих обстоятельствах использование -F вместо --form-string может позволить пользователю обманом заставить curl загрузить файл.

Референт

HTTP-запрос имеет возможность включать информацию о том, какой адрес перенаправил его на фактическую страницу. curl позволяет указать референт, который будет использоваться в командной строке. Это особенно полезно для обмана глупых серверов или CGI-скриптов, которые полагаются на доступность этой информации или содержат определенные данные.

```
curl -e www.coolsite.com http://www.showme.com/
```

Пользовательский агент

HTTP-запрос имеет возможность включать информацию о браузере, который сгенерировал запрос. Curl позволяет указывать его в командной строке. Это особенно полезно для обмана глупых серверов или CGI-скриптов, которые принимают только определенные браузеры.

Пример:

```
curl -A 'Mozilla/3.0 (Win95; I)' http://www.nationsbank.com/
```

Другие распространенные строки:

- Mozilla/3.0 (Win95; I) - Netscape версии 3 для Windows 95
- Mozilla/3.04 (Win95; U) - Netscape версии 3 для Windows 95
- Mozilla/2.02 (OS/2; U) - Netscape версии 2 для OS / 2
- Mozilla/4.04 [en] (X11; U; AIX 4.2; Nav) - Netscape для AIX
- Mozilla/4.05 [en] (X11; U; Linux 2.0.32 i586) - Netscape для Linux

Обратите внимание, что Internet Explorer изо всех сил старается быть совместимым во всех отношениях:

- Mozilla/4.0 (compatible; MSIE 4.01; Windows 95) - MSIE для W95

Mozilla - не единственное возможное имя пользовательского агента:

- Konqueror/1.0 - Клиент для настольного файлового менеджера KDE
- Lynx/2.7.1 libwww-FM/2.14 - Браузер командной строки Lynx

Файлы cookie

Файлы cookie обычно используются веб-серверами для хранения информации о состоянии на стороне клиента. Сервер устанавливает cookie-файлы, отправляя строку ответа в заголовках, которая выглядит следующим образом, Set-Cookie: <data> где часть данных затем обычно содержит набор NAME=VALUE пар (разделенных точками с запятой, ; например NAME1=VALUE1; NAME2=VALUE2;). Сервер также может указать, для какого пути следует использовать файл cookie (указав path=value), когда срок действия файла cookie истекает (expire=DATE), для какого домена его использовать (domain=NAME) и следует ли его использовать только для безопасных подключений (secure).

Если вы получили страницу с сервера, которая содержит заголовок, подобный:

```
Set-Cookie: sessionid=boo123; path="/foo";
```

это означает, что сервер хочет, чтобы эта первая пара передавалась дальше, когда мы получаем что-либо в пути, начинающемся с /foo.

Например, получить страницу, которая хочет, чтобы мое имя было передано в файле cookie:

```
curl -b "name=Daniel" www.sillypage.com
```

Curl также имеет возможность использовать ранее полученные файлы cookie в последующих сеансах. Если вы получаете файлы cookie с сервера и сохраняете их в файле способом, подобным:

```
curl --dump-header headers www.example.com
```

... затем вы можете через секунду подключиться к этому (или другому) сайту, использовать файлы cookie из headers.txt файла, например:

```
curl -b headers.txt www.example.com
```

Хотя сохранение заголовков в файл является рабочим способом хранения файлов cookie, однако он подвержен ошибкам и не является предпочтительным способом сделать это. Вместо этого заставьте curl сохранять входящие файлы cookie, используя хорошо известный формат файлов cookie Netscape, подобный этому:

```
curl -c cookies.txt www.example.com
```

Обратите внимание, что, указав, -b вы включаете механизм использования файлов cookie и с помощью -L вы можете заставить curl следовать за location: (который часто используется в сочетании с файлами cookie). Если сайт отправляет файлы cookie и поле местоположения, вы можете использовать несуществующий файл, чтобы активировать оповещение о файлах cookie, например:

```
curl -L -b empty.txt www.example.com
```

Файл для чтения файлов cookie должен быть отформатирован с использованием обычных HTTP-заголовков или как файл cookie Netscape. Curl определит, какой это тип, на основе содержимого файла. В приведенной выше команде curl проанализирует заголовок и сохранит файлы cookie, полученные отwww.example.com. curl отправит на сервер сохраненные файлы cookie, которые соответствуют запросу, поскольку он следует за местоположением. Этот файл empty.txt может быть несуществующим файлом.

Для чтения и записи файлов cookie из файла cookie Netscape вы можете настроить и -b, и -c использовать один и тот же файл:

```
curl -b cookies.txt -c cookies.txt www.example.com
```

Индикатор прогресса

Индикатор выполнения существует для того, чтобы показывать пользователю, что что-то действительно происходит. Различные поля в выходных данных имеют следующее значение:

```
% Total    % Received % Xferd  Average Speed          Time             Curr.
                               Dload  Upload Total    Current  Left    Speed
0  151M    0 38608    0     0   9406      0  4:41:43  0:00:04  4:41:39  9287
```

Слева направо:

- % - процент завершения всего переноса
- Total - общий размер всего ожидаемого переноса
- % - процент завершения загрузки
- Received - количество загруженных в данный момент байт
- % - процент завершения загрузки
- Xferd - количество загруженных в данный момент байт
- Average Speed Dload - средняя скорость передачи при загрузке
- Average Speed Upload - средняя скорость передачи при загрузке
- Time Total - ожидаемое время для завершения операции
- Time Current - время, прошедшее с момента вызова
- Time Left - ожидаемое время, оставшееся до завершения
- Curr.Speed - средняя скорость передачи за последние 5 секунд (разумеется, первые 5 секунд передачи рассчитаны на меньшее время).

-# Опция отобразит совершенно другой индикатор выполнения, который не требует особых объяснений!

Ограничение скорости

Curl позволяет пользователю устанавливать условия скорости передачи, которые должны соблюдаться для продолжения передачи. С помощью переключателя -y и -Y вы можете заставить curl прервать передачу, если скорость передачи ниже указанного минимального предела в течение указанного времени.

Чтобы заставить curl прервать загрузку, если скорость меньше 3000 байт в секунду в течение 1 минуты, запустите:

```
curl -Y 3000 -y 60 www.far-away-site.com
```

Это можно использовать в сочетании с общим ограничением по времени, так что вышеуказанная операция должна быть выполнена полностью в течение 30 минут:

```
curl -m 1800 -Y 3000 -y 60 www.far-away-site.com
```

Также возможно заставить curl не передавать данные быстрее заданной скорости, что может быть полезно, если вы используете соединение с ограниченной пропускной способностью и не хотите, чтобы ваша передача использовала всю ее (иногда называемую *ограничением пропускной способности*).

Сделайте так, чтобы curl передавал данные не быстрее 10 килобайт в секунду:

```
curl --limit-rate 10K www.far-away-site.com
```

или

```
curl --limit-rate 10240 www.far-away-site.com
```

Или запретить curl загружать данные быстрее, чем 1 мегабайт в секунду:

```
curl -T upload --limit-rate 1M ftp://uploadshereplease.com
```

При использовании `--limit-rate` опции скорость передачи регулируется посекундно, что приведет к тому, что общая скорость передачи станет ниже заданного значения. Иногда, конечно, значительно ниже, если ваша передача останавливается во время периодов.

Файл конфигурации

Curl автоматически пытается прочитать `.curlrc` файл (или `_curlrc` файл в системах Microsoft Windows) из домашней папки пользователя при запуске.

Конфигурационный файл может быть составлен с помощью обычных переключателей командной строки, но вы также можете указать длинные параметры без тире, чтобы сделать его более читаемым. Вы можете разделять опции и параметр пробелами или с помощью = или :. Внутри файла можно использовать комментарии. Если первая буква в строке является #-символом, остальная часть строки обрабатывается как комментарий.

Если вы хотите, чтобы параметр содержал пробелы, вы должны заключить весь параметр в двойные кавычки ("). Внутри этих кавычек вы указываете кавычку как \".

ПРИМЕЧАНИЕ: Вы должны указать параметры и их аргументы в одной строке.

Например, установите тайм-аут по умолчанию и прокси-сервер в файле конфигурации:

```
# We want a 30 minute timeout:
-m 1800
#. .. and we use a proxy for all accesses:
proxy = proxy.our.domain.com:8080
```

Пробелы важны в конце строк, но все пробелы, ведущие к первым символам каждой строки, игнорируются.

Запретите curl читать файл по умолчанию, используя -q в качестве первого параметра командной строки, например:

```
curl -q www.thatsite.com
```

Принудительно используйте curl для получения и отображения локальной страницы справки в случае, если она вызывается без URL, создав конфигурационный файл, подобный:

```
# default url to get
url = "http://help.with.curl.com/curlhelp.html"
```

Вы можете указать другой файл конфигурации для чтения, используя флаг -K/--config . Если вы установите для имени файла конфигурации значение-, он будет считывать конфигурацию из stdin, что может быть удобно, если вы хотите скрыть параметры от отображения в таблицах процессов и т. Д:

```
echo "user = user:passwd" | curl -K - http://that.secret.site.com
```

Дополнительные заголовки

При использовании curl в ваших собственных программах может возникнуть необходимость передавать собственные пользовательские заголовки при получении веб-страницы. Вы можете сделать это с помощью -H флага.

Например, отправьте заголовок X-you-and-me: yes на сервер при получении страницы:

```
curl -H "X-you-and-me: yes" www.love.com
```

Это также может быть полезно, если вы хотите, чтобы curl отправлял текст в заголовке, отличный от обычного. -H Указанный вами заголовок затем заменяет заголовок, который обычно отправляет curl. Если вы заменяете внутренний заголовок пустым, вы предотвращаете отправку этого заголовка. Чтобы предотвратить использование Host: заголовка:

```
curl -H "Host:" www.server.com
```

Имена FTP и путей

Обратите внимание, что при получении файлов с ftp:// URL-адресом указанный путь относится к каталогу, который вы вводите. Чтобы получить файл README из вашего домашнего каталога на вашем ftp-узле, выполните:

```
curl ftp://user:passwd@my.site.com/README
```

Если вам нужен файл README из корневого каталога того же сайта, вам необходимо указать абсолютное имя файла:

```
curl ftp://user:passwd@my.site.com//README
```

(Т.е. с дополнительной косой чертой перед именем файла.)

SFTP и SCP и имена путей

При использовании sftp: и scp: URL-адресов указанное имя пути является абсолютным именем на сервере. Чтобы получить доступ к файлу, относящемуся к домашнему каталогу удаленного пользователя, добавьте к файлу префикс /~/ , например:

```
curl -u $USER sftp://home.example.com/~/.bashrc
```

FTP и брандмауэры

Протокол FTP требует, чтобы одна из вовлеченных сторон открыла второе соединение, как только данные будут готовы к передаче. Есть два способа сделать это.

По умолчанию для curl используется команда PASV, которая заставляет сервер открыть другой порт и ожидать другого подключения, выполняемого клиентом. Это хорошо, если клиент находится за брандмауэром, который не разрешает входящие подключения.

```
curl ftp.download.com
```

Если сервер, например, находится за брандмауэром, который не разрешает подключения по портам, отличным от 21 (или если он просто не поддерживает PASV команду), другой способ сделать это - использовать PORT команду и указать серверу подключиться к клиенту по заданному IP-номеру и порту (в качестве параметров для команды PORT).

-P Флаг для curl поддерживает несколько различных опций. У вашего компьютера может быть несколько IP-адресов и / или сетевых интерфейсов, и curl позволяет вам выбрать, какой из них использовать. Адрес по умолчанию также может быть использован:

```
curl -P - ftp.download.com
```

Загружайте с помощьюPORT, но используйте IP-адрес нашего leO интерфейса (это не работает в Windows):

```
curl -P leO ftp.download.com
```

Загружайте cPORT, но используйте 192.168.0.10 в качестве нашего IP-адреса для использования:

```
curl -P 192.168.0.10 ftp.download.com
```

Сетевой интерфейс

Получить веб-страницу с сервера, используя указанный порт для интерфейса:

```
curl --interface eth0:1 http://www.example.com/
```

или

```
curl --interface 192.168.1.10 http://www.example.com/
```

HTTPS

Для безопасного HTTP требуется, чтобы при создании curl была установлена и использовалась библиотека TLS. Если это сделано, curl способен извлекать и публиковать документы, используя протокол HTTPS.

Пример:

```
curl https://www.secure-site.com
```

curl также способен использовать клиентские сертификаты для получения / публикации файлов с сайтов, которым требуются действительные сертификаты. Единственным недостатком является то, что сертификат должен быть в формате PEM. PEM - это стандартный и открытый формат для хранения сертификатов, но он не используется наиболее часто используемыми браузерами. Если вы хотите, чтобы curl использовал сертификаты, которые вы используете в своем любимом браузере, вам может потребоваться загрузить / скомпилировать конвертер, который может конвертировать сертификаты в формате вашего браузера в формат PEM.

Пример автоматического получения документа с использованием сертификата с персональным паролем:

```
curl -E /path/to/cert.pem:password https://secure.site.com/
```

Если вы пренебрегаете указанием пароля в командной строке, вам будет предложено ввести правильный пароль, прежде чем можно будет получить какие-либо данные.

Многие старые HTTPS-серверы имеют проблемы с определенными версиями SSL или TLS, которые используют более новые версии OpenSSL и т.д., Поэтому Иногда полезно указать, какую версию TLS должен использовать curl.:

```
curl --tlv1.0 https://secure.site.com/
```

В противном случае curl попытается использовать разумную версию TLS по умолчанию.

Возобновление передачи файлов

Чтобы продолжить передачу файлов с того места, где она была ранее прервана, curl поддерживает возобновление при загрузке по HTTP, а также при загрузке по FTP.

Продолжить загрузку документа:

```
curl -C - -o file ftp://ftp.server.com/path/file
```

Продолжить загрузку документа:

```
curl -C - -T file ftp://ftp.server.com/path/file
```

Продолжить загрузку документа с веб-сервера

```
curl -C - -o file http://www.server.com/
```

Временные условия

HTTP позволяет клиенту указать условие времени для документа, который он запрашивает. Это If-Modified-Since или If-Unmodified-Since. curl позволяет указывать их с помощью флага -z/--time-cond.

Например, вы можете легко выполнить загрузку, которая выполняется только в том случае, если удаленный файл более новый, чем локальная копия. Это было бы сделано как:

```
curl -z local.html http://remote.server.com/remote.html
```

Или вы можете загрузить файл, только если локальный файл новее удаленного. Сделайте это, добавив перед строкой даты -, как в:

```
curl -z -local.html http://remote.server.com/remote.html
```

В качестве условия можно указать дату в виде обычного текста. Попросите curl загружать файл, только если он был обновлен с 12 января 2012 года:

```
curl -z "Jan 12 2012" http://remote.server.com/remote.html
```

curl поддерживает широкий диапазон форматов даты. Вы всегда делаете проверку даты наоборот, добавляя к ней тире (-).

ДИКТ

Для развлечения попробуйте

```
curl dict://dict.org/m:curl
curl dict://dict.org/d:heisenbug:jargon
curl dict://dict.org/d:daniel:gcide
```

Псевдонимами для m являются match и find, а псевдонимами для d являются define и lookup. Например,

```
curl dict://dict.org/find:curl
```

Команды, которые нарушают URL-описание RFC (но не протокол DICT), являются

```
curl dict://dict.org/show:db
curl dict://dict.org/show:strat
```

Поддержка аутентификации по-прежнему отсутствует

LDAP

Если вы установили библиотеку OpenLDAP, curl может воспользоваться ее преимуществами и предложить ldap:// поддержку. В Windows curl по умолчанию будет использовать WinLDAP из Platform SDK.

Версия протокола по умолчанию, используемая curl, - LDAP версии 3. Версия 2 будет использоваться в качестве запасного механизма на случай, если не удастся подключиться к версии 3.

LDAP - сложная вещь, и написание запроса LDAP - непростая задача. Ознакомьтесь с точным описанием синтаксиса в другом месте. Одним из таких мест может быть: RFC 2255, формат URL LDAP

Чтобы показать вам пример, вот как получить всех пользователей с сервера LDAP, в адресе электронной почты которого указан определенный поддомен:

```
curl -B "ldap://ldap.frontec.se/o=frontec??sub?mail=*sth.frontec.se"
```

Вы также можете использовать аутентификацию при доступе к каталогу LDAP:

```
curl -u user:passwd "ldap://ldap.frontec.se/o=frontec??sub?mail=*"
curl "ldap://user:passwd@ldap.frontec.se/o=frontec??sub?mail=*"
```

По умолчанию, если пользователь и пароль указаны, OpenLDAP / WinLDAP будут использовать базовую аутентификацию. В Windows вы можете управлять этим поведением, указав один из --basic, --ntlm или --digest параметров в командной строке curl

```
curl --ntlm "ldap://user:passwd@ldap.frontec.se/o=frontec??sub?mail=*"
```

В Windows, если пользователь / пароль не указаны, механизм автоматического согласования будет использоваться с текущими учетными данными для входа (SSPI / SPNEGO).

Переменные среды

Curl считывает и понимает следующие переменные среды:

```
http_proxy, HTTPS_PROXY, FTP_PROXY
```

Они должны быть установлены для прокси, зависящих от протокола. Общий прокси-сервер должен быть установлен с

```
ALL_PROXY
```

Разделенный запятыми список имен хостов, которые не должны проходить через какой-либо прокси, указан в (только звездочка, * соответствует всем хостам)

```
NO_PROXY
```

Если имя хоста соответствует одной из этих строк или хост находится в домене одной из этих строк, транзакции с этим узлом не будут выполняться через прокси. Когда используется домен, он должен начинаться с точки. Пользователь может указать, что обаwww.example.com и foo.example.com не следует использовать прокси, установив NO_PROXY значение .example.com. Указав полное имя, вы можете исключить конкретные имена хостов, поэтому, чтобы заставить www.example.com не использовать прокси, но все равно foo.example.com это делать, установите NO_PROXY значение www.example.com.

Использование флага -x/--proxy переопределяет переменные среды.

Netrc

Unix представила .netrc концепцию давным-давно. Это способ для пользователя указать имя и пароль для часто посещаемых FTP-сайтов в файле, чтобы вам не приходилось вводить их при каждом посещении этих сайтов. Вы понимаете, что это большой риск для безопасности, если кто-то другой получит ваши

пароли, поэтому большинство программ Unix не будут читать этот файл, если только он не доступен для чтения только вам (curl, впрочем, все равно).

Curl поддерживает .netrc файлы, если указано иное (используя опции -n/--netrc и --netrc-optional). Это не ограничивается только FTP, поэтому curl может использовать его для всех протоколов, где используется аутентификация.

Простой .netrc файл может выглядеть примерно так:

```
machine curl.se login iamdaniel password mysecret
```

Пользовательский вывод

Чтобы программисты сценариев могли лучше узнавать о ходе работы над curl, была введена опция -w/--write-out . Используя это, вы можете указать, какую информацию из предыдущей передачи вы хотите извлечь.

Для отображения количества загруженных байт вместе с некоторым текстом и конечной новой строкой:

```
curl -w 'We downloaded %{size_download} bytes\n' www.download.com
```

Передача по FTP Kerberos

Curl поддерживает kerberos4 и kerberos5 / GSSAPI для передачи по FTP. Вам необходимо установить пакет kerberos и использовать его во время сборки curl, чтобы он был доступен.

Сначала получите krb-ticket обычным способом, например, с помощью инструмента kinit/kauth . Затем используйте curl способом, подобным:

```
curl --krb private ftp://krb4site.com -u username:fakepwd
```

Пароль на -u коммутаторе не используется, но пустой пароль заставит curl запросить его, а вы уже ввели реальный пароль для kinit/kauth.

TELNET

Поддержка curl telnet является базовой и простой в использовании. Curl передает все данные, переданные ему по стандартному интерфейсу, на удаленный сервер. Подключитесь к удаленному серверу telnet, используя командную строку, аналогичную:

```
curl telnet://remote.server.com
```

И введите данные для передачи на сервер в stdin. Результат будет отправлен в стандартный вывод или в файл, который вы укажете с помощью -o.

Возможно, вы захотите, чтобы опция -N/--no-buffer отключала буферизованный вывод для медленных подключений или чего-то подобного.

Передайте параметры для согласования протокола telnet, используя -t параметр. Чтобы сообщить серверу, что мы используем терминал vt100, попробуйте что-то вроде:

```
curl -tTTYPE=vt100 telnet://remote.server.com
```

Другие интересные варианты для этого -t включают:

- XDISPLOC=<X display> Задает местоположение отображения X.
- NEW_ENV=<var,val> Задает переменную среды.

ПРИМЕЧАНИЕ: Протокол telnet не определяет какой-либо способ входа с указанным пользователем и паролем, поэтому curl не может сделать это автоматически. Для этого вам нужно отследить, когда получено приглашение для входа, и соответствующим образом отправить имя пользователя и пароль.

Постоянные соединения

Указание нескольких файлов в одной командной строке заставит curl перенести их все, один за другим в указанном порядке.

libcurl попытается использовать постоянные соединения для передач, чтобы вторая передача на тот же хост могла использовать то же соединение, которое уже было инициировано и оставалось открытым при предыдущей передаче. Это значительно сокращает время соединения для всех операций, кроме первой передачи, и значительно улучшает использование сети.

Обратите внимание, что curl не может использовать постоянные соединения для передач, которые используются в последующих вызовах curl. Попробуйте поместить как можно больше URL-адресов в одну и ту же командную строку, если они используют один и тот же хост, так как это ускорит передачу. Если вы используете HTTP-прокси для передачи файлов, практически все передачи будут постоянными.

Несколько передач с помощью одной командной строки

Как упоминалось выше, вы можете загружать несколько файлов с помощью одной командной строки, просто добавив дополнительные URL-адреса. Если вы хотите, чтобы они сохранялись в локальный файл, а не просто печатались в стандартный вывод, вам нужно добавить один параметр сохранения для каждого указанного вами URL. Обратите внимание, что это также относится к -O опции (но не --remote-name-all).

Например: возьмите два файла и используйте -O для первого и пользовательское имя файла для второго:

```
curl -O http://url.com/file.txt ftp://ftp.com/moo.exe -o moo.jpg
```

Вы также можете загрузить несколько файлов аналогичным образом:

```
curl -T local1 ftp://ftp.com/moo.exe -T local2 ftp://ftp.com/moo2.txt
```

IPv6

curl подключится к серверу с IPv6, когда поиск по хосту вернет адрес IPv6, и вернется к IPv4, если соединение не удастся. Опции --ipv4 и --ipv6 могут указывать, какой адрес использовать, когда оба доступны. Адреса IPv6 также могут быть указаны непосредственно в URL-адресах, используя синтаксис:

```
http://[2001:1890:1112:1::20]/overview.html
```

При использовании этого стиля необходимо указать -g опцию, чтобы запретить curl интерпретировать квадратные скобки как специальные символы с глобулированием. Также могут использоваться локальные адреса ссылок и сайтов, включая идентификатор области, такой как fe80::1234%1, но часть области должна быть числовой или соответствовать существующему сетевому интерфейсу в Linux, а символ процента должен быть экранированным URL. Предыдущий пример в URL-адресе SFTP может выглядеть следующим образом:

```
sftp://[fe80::1234%251]/
```

Адреса IPv6, указанные не в URL-адресах (например, для --proxy, --interface или --ftp-port опций), не должны быть закодированы в URL.

Списки рассылки

Для вашего удобства у нас есть несколько открытых списков рассылки, в которых обсуждается curl, его разработка и все, что с этим связано. Получите всю информацию на https://curl.se/mail.

Пожалуйста, направляйте вопросы curl, пожелания к функциям и сообщения о неполадках в один из этих списков рассылки вместо рассылки кому-либо другому.

Доступные списки включают:

curl-users

Пользователи инструмента командной строки. Как его использовать, что не работает, новые функции, связанные инструменты, вопросы, новости, установки, компиляции, запуск, перенос и т.д.

curl-library

Разработчики, использующие или разрабатывающие libcurl. Ошибки, расширения, улучшения.

curl-announce

Низкий трафик. Получает только объявления о новых общедоступных версиях. В худшем случае получается что-то вроде одного или двух писем в месяц, но обычно только одно письмо раз в два месяца.

curl-and-php

Использование функций curl в PHP. Все скручивается под углом PHP. Или PHP с углом скручивания.

curl-and-python

Хакеры Python, использующие curl с или без привязки python к pycurl.