УТВЕРЖДЕН
СЕИУ.00009-05 34 12 - ЛУ

СРЕДСТВО КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ
«МагПро КриптоПакет» 4.0

Средство удалённого доступа к сетевым ресурсам curl.
Руководство по использованию

СЕИУ.00009-05 34 12
Листов 62

Литера О

2022

Аннотация

Настоящий документ содержит руководство по использованию средства удалённого доступа к сетевым ресурсам curl из состава «МагПро КриптоПакет» 4.0.

Содержание

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

1 НАЗНАЧЕНИЕ ПРОГРАММЫ

Утилита curl -– это командно-строчная утилита для передачи данных с сервера или на сервер с использованием одного из поддерживаемых протоколов (DICT, FILE, FTP, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SMB, SMBS, SMTP, SMTPS, TELNET и TFTP). Утилита предназначена для работы без взаимодействия с пользователем. Утилита curl реализует такие механизмы, как поддержка прокси, аутентификация пользователя, загрузка FTP, HTTP-сообщение, TLS-соединения, файлы cookie, передача файлов.

Утилита curl -– это составная часть исполнения 3 (соответствует классу КС1) и исполнения 4 (соответствует классу КС2) СКЗИ «МагПро КриптоПакет» 4.0.

Утилита curl является функционально законченным изделием.

Все криптографические преобразования и протоколы основаны, в первую очередь, на алгоритмах ГОСТ. Алгоритмы шифрования, расшифровывания и имитозащиты соответствуют ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015, алгоритмы вычисления хэш-функции соответствуют ГОСТ Р 34.11-2012, алгоритмы создания и проверки ЭП соответствуют ГОСТ Р 34.10-2012. Для обеспечения совместимости с предыдущими версиями СКЗИ сохранена также поддержка устаревших алгоритмов ГОСТ 28147-89, ГОСТ Р 34.10-2001 и ГОСТ Р 34.11-94. В большинстве случаев алгоритмы ГОСТ используются по умолчанию.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

2 УСЛОВИЯ РАБОТЫ ПРОГРАММЫ

Утилита curl предназначена для работы в следующих операционных системах:

Windows 8.1/10;
Windows Server 2012/2016/2019;
Debian GNU/Linux 9(stretch)/10(buster)/11(bullseye);
Ubuntu 14.04, 16.04, 18.04, 20.04;
Linux Mint 19.x, 20.x, Linux Mint Debian Edition 4;
RedHat Enterprise Linux 7, 8;
CentOS 7, 8;
SUSE Linux 12, 15;
OpenSUSE 15.1, 15.2;
EMIAS OS 1.0, 2.0;
Дистрибутивы Альт на базе платформ 8 и 9, включая Альт Сервер, Альт Рабочая станция, Альт Рабочая станция К, Альт Образование, Альт 8 СП, Sipmly Linux;
МСВСфера Сервер 7.3, МСВСфера АРМ 7.3;
Гослинукс IC6;
РЕД ОС 7.2, 7.3;
Rosa Enterprise Desktop (RED) X4;
Rosa Enterprise Linux Server (RELS) 7.3;
Rosa Enterprise Linux Desktop (RELD) 7.3;
РОСА КОБАЛЬТ;
Astra Linux Special Edition Смоленск 1.6 aka исп.1, 1.7;
Astra Linux Special Edition Новороссийск;
Astra Linux Common Edition 2.12;
Numa Edge 1.0;
FreeBSD 12.x, 13.x;
MacOS 10.15, 11;
Sun Solaris 10, 11;
OpenWRT 19.07, 21.02.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

3 ФУНКЦИИ ПРОГРАММЫ

Утилита curl предоставляет возможность использования протокола TLS с использованием российских криптонаборов, определённых рекомендациями ТК26, в различных сетевых протоколах прикладного уровня.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

4 Обеспечение информационной безопасности при использовании «МагПро КриптоПакет» 4.0

Надежная криптографическая защита данных при использовании «МагПро КриптоПакет» 4.0 обеспечивается только в том случае, если эксплуатация «МагПро КриптоПакет» 4.0 осуществляется в строгом соответствии с требованиями документа «Средство криптографической защиты информации «МагПро КриптоПакет» 4.0. Правила пользования» (СЕИУ.00009–05 94).

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

5 УСТАНОВКА И НАСТРОЙКА ПРОГРАММЫ

Программа устанавливается и настраивается автоматически при установке утилиты openssl (см. «Средство криптографической защиты информации «МагПро КриптоПакет» 4.0. Утилита openssl. Руководство по использованию», СЕИУ.00009-05 34 03).

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

6 ИСПОЛЬЗОВАНИЕ ПРОГРАММЫ

6.1 Формат запуска программы

Формат запуска программы:

```
curl [опции / URL]
```

6.2 URL

Синтаксис URL-адреса зависит от протокола, подробности описаны в RFC 3986.

Утилите можно указать несколько URL-адресов. Наборы однотипных адресов можно указывать, написав варианты в фигурных скобках, например:

```
"http://site.{one,two,three}.com"
```

или буквенно-цифровую последовательность, используя [], например:

```
"ftp://ftp.example.com/file[1-100].txt"
"ftp://ftp.example.com/file[001-100].txt"
"ftp://ftp.example.com/file[a-z].txt"
```

Вложенные последовательности не поддерживаются, но можно использовать несколько рядом друг с другом:

```
"http://example.com/archive[1996-1999]/vol[1-4]/part{a,b,c}.html"
```

В командной строке можно указать любое количество URL-адресов, они будут извлекаться последовательно в указанном порядке. Параметры командной строки и URL-адреса можно указывать в любом порядке.

Вы можете указать счетчик шагов для диапазонов, чтобы получить каждое N-е число или букву:

```
"http://example.com/file[1-100:10].txt"
"http://example.com/file[a-z:2].txt"
```

При использовании последовательностей [] или при вызове из командной строки вам, вероятно, придется заключить полный URL-адрес в двойные кавычки, чтобы оболочка не мешала ему. Это также относится к другим символам, рассматриваемым как особые, например, '&', '?' и '*'.

Индекс зоны IPv6 в URL-адресе следует указывать с экранированным знаком процента и именем интерфейса, например

```
"http://[fe80::3%25eth0]/"
```

Если вы укажете URL без префикса protocol://, утилита попытается угадать, какой протокол вам может понадобиться. По умолчанию она будет использовать HTTP, но попробует другие протоколы, отталкиваясь от часто используемых префиксов имени хоста. Например, для имен хостов, начинающихся с «ftp», утилита будет использовать протокол FTP.

Утилита curl сделает все возможное, чтобы обеспечить доступ к тому, что вы ему передали в качестве URL-адреса. Однако она никоим образом не будет пытаться проверить его синтаксическую правильность, напротив, она будет очень либеральна по отношению к полученному адресу.

Утилита curl будет пытаться повторно использовать соединения для передачи нескольких файлов, поэтому получение большого количества файлов с одного сервера не приведет к многократной установке соединения, что увеличивает скорость скачивания. Конечно, это делается

только для файлов, указанных в одной командной строке, и не может использоваться между отдельными вызовами утилиты.

6.3 Вывод программы

Если не указано иное, утилита curl выводит полученные данные в stdout. Опции -o, --output или -O, --remote-name позволяют сохранить эти данные в файл. Если в командной строке утилиты указано несколько URL-адресов, требуется также указать несколько имён файлов для сохранения данных.

Утилита curl не анализирует и не «понимает» контент, который он получает, он не выполняет кодирования или декодирования, если это явно не указано в специальных параметрах командной строки.

6.4 Протоколы

Утилита curl поддерживает множество протоколов:

DICT Позволяет искать слова с помощью онлайн-словарей.

FILE Чтение или запись локальных файлов. Утилита curl не поддерживает удаленный доступ к file:// URL, но при работе в Microsoft Windows с использованием собственного подхода UNC будет работать.

FTP Протокол передачи данных поддерживается без использования TLS.

GOPHER Извлечение файлов.

HTTP(S) Утилита curl поддерживает HTTP версию 0.9, 1.0, 1.1, 2 и 3 в зависимости от параметров сборки и правильных параметров командной строки.

IMAP(S) Используя протокол чтения почты, утилита curl может «загружать» электронные письма для вас как с использованием, так и без использования TLS.

LDAP(S) Утилита curl может выполнять поиск в службе каталогов как с TLS, так и без него.

MQTT Утилита curl поддерживает MQTT версии 3. Скачивание по MQTT равнозначно «подписке» на тему, в то время как загрузка на сервер равнозначна «публикации» по теме. Поддержка MQTT является экспериментальной, а MQTT на основе TLS не поддерживается.

POP3(S) Загрузка с сервера pop3 означает получение почты, может работать как с использованием, так и без использования TLS.

RTMP(S) Протокол обмена сообщениями в реальном времени в основном используется для сервера потокового мультимедиа, и утилита curl может его загрузить.

RTSP Утилита curl поддерживает загрузку RTSP 1.0.

SMB(S) Утилита curl поддерживает SMB версии 1

SMTP(S) Загрузка содержимого на SMTP-сервер означает отправку электронной почты, может работать как с TLS, так и без него.

TELNET Указание утилите curl получить URL-адрес telnet запускает интерактивный сеанс, в котором он отправляет то, что он читает в stdin, и выводит на stdout то, что отправляет сервер.

6.5 Прогресс-индикация

Утилита curl обычно отображает индикатор прогресса во время операций, указывая объем передаваемых данных, скорость передачи, предполагаемое оставшееся время и т. Д. Индикатор прогресса отображает количество байтов, а скорость-в байтах в секунду. Суффиксы (k, M, G, T, P) основаны на 1024, например, 1k-это 1024 байта. 1M-это 1048576 байт.

Утилита curl отображает эти данные на терминале по умолчанию, поэтому если вы вызываете утилиту curl для выполнения операции, и он собирается записать данные на терминал, он отключает прогресс-индикацию, так как в противном случае она испортит вывод данных ответа.

Если вам нужен индикатор выполнения запросов HTTP POST или PUT, вам нужно перенаправить вывод ответа в файл, используя shell redirect (>), -o, --output или аналогичный.

Это не относится к загрузке по FTP, поскольку эта операция не выдает никаких данных ответа на терминал.

Если вы предпочитаете индикатор прогресса вместо обычного счетчика, используйте опцию -#, --progress-bar. Вы также можете полностью отключить индикатор прогресса с помощью опции -s, --silent.

6.6 Опции программы

Опции указываются с одним или двумя тире в начале. Многим опциям требуются дополнительные параметры.

Короткая форма опций с одним тире, например -d, может использоваться с пробелом или без пробела перед параметром опции, при этом пробел является рекомендуемым разделителем. При применении длинной формы, например --data, пробел необходим.

Опции в короткой версии, которые не нуждаются в каких-либо дополнительных значениях, могут использоваться непосредственно рядом друг с другом, например, вы можете указать все параметры -O,- L и -v сразу, как -OLv.

Обычно все логические(булевы) опции включены указанием типа --option, но возможно их отключить с помощью --no-option. То есть используется точно такое же имя опции с префиксом "no-". В этой инструкции в основном описываются опции указанные --option.

--abstract-unix-socket <путь>

(HTTP) Подключает через абстрактный доменный сокет Unix вместо использования сети. Примечание: netstat показывает путь абстрактного сокета с префиксом '@', однако аргумент <путь> не должен начинаться с данного символа.

--alt-svc <имя файла>

(HTTPS) ПРЕДУПРЕЖДЕНИЕ: этот параметр является экспериментальным. Не используйте в производстве.

Этот параметр включает синтаксический анализатор alt-svc в утилите curl. Если имя файла указывает на существующий файл кэша alt-svc, он будет использоваться. После завершения анализа кэш будет сохранён в указанный файл, если он был изменен.

Укажите имя файла пустым "" (нулевая длина), чтобы избежать загрузки/сохранения и заставить утилиту curl просто обрабатывать кэш в памяти.

Если эта опция используется несколько раз, утилита загрузит содержимое из всех файлов, но последний будет использоваться для сохранения.

--anyauth

(HTTP) Данная опция предписывает утилите curl самостоятельно определить метод

аутентификации и использовать наиболее безопасный, который поддерживает удаленный сайт. Это делается путем первичного выполнения запроса и проверки заголовков ответов, таким образом возможно дополнительное сетевое обращение. Такой метод используется вместо установки определенного метода аутентификации, который вы можете сделать с помощью --basic, --digest и --ntlm.
Использование --anyauth не рекомендуется, если вы выполняете загрузку из stdin, так как для этого может потребоваться отправить данные дважды, а затем клиент должен иметь возможность перемотать назад. Если при загрузке из stdin возникнет такая необходимость, операция загрузки завершится неудачно.
Используется вместе с -u, --user.
См. также --proxy-anyauth, --basic и --digest.

-a, --append

(FTP) При использовании в загрузке эта опция заставляет утилиту curl добавлять к целевому файлу вместо его перезаписи. Если файл не существует, он будет создан.

--aws-sigv4 <поставщик1[:поставщик2[:регион[:служба]]]>

Использует аутентификацию подписи AWS V4 при передаче.
Аргумент <поставщик> это строка, используемая алгоритмом при создании заголовков исходящей проверки подлинности.
Аргумент <регион> это строка, указывающая на географическую область коллекции ресурсов (код региона), если имя региона не указано в конечной точке.
Аргумент <служба> это строка, указывающая на функцию, предоставляемую облаком (код службы), когда имя службы не указано в конечной точке.

--basic

(HTTP) Указывает использовать базовую аутентификацию HTTP с удаленным хостом. Это значение по умолчанию, и этот параметр обычно бессмыслен, если вы не используете его для переопределения ранее установленного параметра, который устанавливает другой метод проверки подлинности (например, --ntlm или --digest).
Используется вместе с -u, --user.
См. также --proxy-basic.

--cacert <файл>

(TLS) Указывает использовать данный файл сертификата УЦ для проверки сертификата сервера, к которому выполняется подключение. Файл может содержать несколько сертификатов УЦ. Сертификаты должны быть в формате PEM.
Если параметр не указан, используются умолчания, описанные ниже.
Утилита curl распознает переменную среды с именем `CURL_CA_BUNDLE`, если она установлена, и использует данный путь в качестве пути к файлу сертификатов УЦ.
Версия утилиты для Windows автоматически будет искать файл сертификатов CA с именем curl-ca-bundle.crt в том же каталоге, что и curl.exe, или в текущем рабочем каталоге, или в любой папке указанной в переменной <PATH>.
Если эта опция используется несколько раз, применяться будет последняя.

--capath <каталог>

(TLS) Предписывает использовать указанный каталог сертификатов для проверки сертификата сервера, к которому выполняется подключение. Можно указать несколько путей, разделив их с помощью `":"` (например, `"путь1:путь2:путь3"`). Сертификаты должны быть в формате PEM и каталог должен быть обработан с помощью утилиты c_rehash.
Использование --capath может позволить создавать TLS-соединения намного эффектив-

нее, чем использование --cacert, если файл --cacert содержит много сертификатов УЦ.
Если эта опция используется, значение capath по умолчанию будет игнорироваться, а если она используется несколько раз, будет использоваться последняя.

--cert-status

(TLS) Указывает проверить статус сертификата сервера по протоколу OCSP.
Если эта опция включена и сервер отправляет недопустимый (например, с истекшим сроком действия) ответ, если в ответе указано, что сертификат сервера был отозван или вообще не получен ответ, проверка завершается неудачей.

--cert-type <type>

(TLS) Формат сертификата клиента. Допустимые форматы: PEM, DER, ENG и P12. Если не указано, предполагается PEM.
Если эта опция используется несколько раз, применяться будет последняя.
См. также -E, --cert, --key and --key-type.

-E, --cert <сертификат[:пароль]>

(TLS) Предписывает использовать указанный файл сертификата клиента при получении файла с HTTPS или другим протоколом на основе TLS. Обратите внимание, что в этом параметре может быть указан файл «сертификата», который объединяет закрытый ключ и сертификат клиента! Чтобы указать закрытый ключ в отдельном файле, используйте параметр --key.
Если эта опция используется несколько раз, применяться будет последняя.
См. также --cert-type, --key и --key-type.

--ciphers <список криптонаборов>

(TLS) Указывает, какие криптонаборы использовать в соединении TLS версий 1.2. Для TLS версии 1.3 следует использовать опцию --tls13-ciphers.
Поддерживаются следующие российские криптонаборы GOST2012-MAGMA-MAGMAOMAC и GOST2012-KUZNYECHIK-KUZNYECHIKOMAC, для совместимости со старыми СКЗИ допустимо также использовать криптонабор GOST2012-GOST8912-GOST8912.
Если эта опция используется несколько раз, применяться будет последняя.

--compressed

(HTTP) Запросит сжатый ответ, используя один из поддерживаемых алгоритмов, и автоматически распакует содержимое. Заголовки не изменяются.
Если эта опция используется и сервер отправляет неподдерживаемую кодировку, утилита сообщит об ошибке.

-K, --config <файл>

Указывает текстовый файл для чтения аргументов утилиты curl. Аргументы командной строки, найденные в текстовом файле, будут использоваться так, как если бы они были предоставлены в командной строке.
Опции и их параметры должны быть указаны в одной строке файла, разделенной пробелом, двоеточием или знаком равенства. Длинные имена опций могут быть дополнительно заданы в файле конфигурации без начальных двойных тире, и если это так, символы двоеточия или равенства могут использоваться в качестве разделителей. Если опция указана с одним или двумя тире, между опцией и её параметром не должно стоять двоеточия или символа равенства.
Если параметр содержит пробелы (или начинается с : или =), то он должен быть заклю-

чен в кавычки. В двойных кавычках доступны следующие escape-последовательности: \\,\", \t, \n, \r и \v. Обратная косая черта, предшествующая любой другой букве, игнорируется. Если первый столбец строки конфигурации содержит символ '#', остальная часть строки будет рассматриваться как комментарий. В файле конфигурации каждая опция записывается на отдельной строке.
Укажите имя файла в -K, --config как '-', чтобы утилита прочитала файл из stdin.
Для того, чтобы указать URL-адрес в файле конфигурации, необходимо указать его с помощью параметра --url, а не просто написать URL-адрес в отдельной строке. Таким образом, это может выглядеть примерно так:
url = "https://curl.se/docs/"
При вызове, утилита curl (если не используется -q, --disable) проверяет наличие файла конфигурации по умолчанию и использует его, если он найден. Файл конфигурации по умолчанию проверяется в следующем порядке:

1. использует переменную среды CURL_HOME, если она установлена;
2. использует переменную среды XDG_CONFIG_HOME, если она установлена;
3. использует переменную HOME среды, если она установлена;
4. не windows: использует getpwuid для поиска домашнего каталога;
5. Windows: использует APPDATA, если установлено;
6. Windows: использует USERPROFILEApplication Data, если установлено;
7. в Windows, если в домашнем каталоге нет файла .curlrc, он проверяет наличие файла в том же каталоге, в котором размещен исполняемый файл curl. В Unix-подобных системах он просто попытается загрузить .curlrc из определенного домашнего каталога.

```
# --- Пример файла ---
# это комментарий
url = "example.com"
output = "curlhere.html"
user-agent = "superagent/1.0"

# и еще один URL
url = "example.com/docs/manpage.html"
-O
referer = "http://nowhereatall.example.com/"
# --- Конец примера файла ---
```

Эта опция может быть использована несколько раз для загрузки нескольких файлов конфигурации.

--connect-timeout <секунды>

Максимальное время в секундах, которое даётся curl для попытки соединения. Это ограничение распространяется только на фазу соединения, поэтому, если утилита curl подключится в течение заданного периода, соединение продолжится, если нет, прервётся. Эта опция принимает десятичные значения.
Если эта опция используется несколько раз, применяться будет последняя.
См.также -m, --max-time.

--connect-to <ХОСТ1:ПОРТ1:ХОСТ2:ПОРТ2>

При запросе к паре ХОСТ1:ПОРТ1 подключение произойдёт к паре ХОСТ2:ПОРТ2. Эта опция подходит для направления запросов на определенный сервер, например, на опреде-

ленный узел кластера в кластере серверов. Эта опция используется только для установления сетевого подключения. Это НЕ влияет на имя хоста/порт, который используется для TLS (например, SNI, проверка сертификата) или для протоколов приложений. <ХОСТ1> и <ПОРТ1> могут быть пустой строкой, означающей "любой хост/порт". <ХОСТ2> и <ПОРТ2> также могут быть пустой строкой, означающей "использовать исходный хост/порт запроса".
<Хост>, указанный в этом параметре, сравнивается как строка, поэтому он должен соответствовать имени, используемому в URL-адресе запроса. Он может быть либо числовым, например `"127.0.0.1"`, либо полным именем хоста, например "example.org".
Этот параметр можно использовать много раз, чтобы добавить множество правил подключения.
См.также --resolve и -H, --header.

-C, --continue-at <смещение>

Продолжить/возобновить предыдущую передачу файлов с заданным смещением. <Смещение> - это точное количество байтов, которые будут пропущены, считая от начала исходного файла. При загрузке файла на сервер не будет использована FTP-команда SIZE.
Опция `"-C -"` используется, чтобы утилита автоматически находила где/как возобновить передачу. Она использует полученные выходные/входные файлы, чтобы выяснить это.
Если эта опция используется несколько раз, применяться будет последняя.
См. также -r, --range.

-c, --cookie-jar <файл>

(HTTP) Указывает, в какой файл будут записаны все файлы cookie после завершения операции. Утилита записывает все файлы cookie из своего хранилища в памяти в данный файл в конце операций. Если файлы cookie неизвестны, никакие данные не будут записаны. Файл будет записан с использованием формата файлов cookie Netscape. Если вместо имени файла стоит дефис `"-"`, файлы cookie будут записаны в stdout.
Эта опция командной строки активирует механизм файлов cookie, который позволяет утилите записывать и использовать файлы cookie. Другой способ активировать его - использовать опцию -b, --cookie.
Если файл cookie не может быть создан или записан, утилита не завершится ошибкой или даже не сообщит об ошибке четко. При использовании -v, --verbose появится предупреждение, но это единственная видимая обратная связь, которую вы получите об этой, возможно, критичной ситуации.
Если эта опция используется несколько раз, будет применяться последнее указанное имя файла.

-b, --cookie <данные|файл>

(HTTP) Передаёт данные на HTTP-сервер в заголовке файла cookie. Предположительно, это данные, ранее полученные с сервера в строке `"Set-Cookie:"`. Данные должны быть в формате `"ИМЯ1=ЗНАЧЕНИЕ1; ИМЯ2=ЗНАЧЕНИЕ2"`.
Если в аргументе нет символа '=', то он используется, как имя файла для чтения ранее сохраненного файла cookie. Эта опция также активирует механизм файлов cookie, который заставит утилиту записывать входящие файлы cookie, что может быть удобно, если вы используете его в сочетании с опцией -L, --location или выполняете несколько передач URL-адресов в одном и том же вызове. Если именем файла является знак минус (`"-"`),

утилита curl будет считывать содержимое из stdin.
Формат файла для чтения файлов cookie должен быть простым HTTP-заголовком(стиль Set-Cookie) или cookie Netscape/Mozilla.
Файл, указанный с помощью -b, --cookie, используется только в качестве входных данных. Файлы cookie не будут записываться в файл. Для хранения файлов cookie используйте опцию -c, --cookie-jar.
Если вы используете формат файла Set-Cookie и не указываете домен, cookie не отправляется, так как домен никогда не будет совпадать. Чтобы решить эту проблему, установите домен в строке Set-Cookie (при этом будут включены поддомены) или, что лучше, используйте формат Netscape.
Эту опцию можно использовать несколько раз.
Пользователи очень часто хотят одновременно читать файлы cookie из файла и записывать обновленные файлы cookie обратно в файл, поэтому использование -b, --cookie, и -c, --cookie-jar в одной командной строке является обычным делом.

--create-dirs

При использовании в сочетании с параметром -o, --output утилита создаст необходимую иерархию локальных каталогов по мере необходимости. Эта опция создает каталоги, упомянутые с параметром -o, --output, и больше ничего. Если в опции --output имя файла не каталог или если упомянутый каталог уже существуют, ничего не будет создано.
Созданные каталоги создаются в режиме 0750 в файловых системах в стиле unix.
Чтобы создать удаленные каталоги при использовании FTP, попробуйте --ftp-create-dirs.

--create-file-mode <режим>

(FILE) Когда утилита используется для удаленного создания файлов с использованием одного из поддерживаемых протоколов, этот параметр позволяет пользователю установить, какой <режим> должен быть установлен для файла во время создания (по умолчанию 0644).
Эта опция принимает восьмеричное число в качестве аргумента.
См. также --ftp-create-dirs.

--crlf

(FTP SMTP) Преобразует LF в CRLF при загрузке. Полезно для MVS (OS/390).

--crlfile <файл>

(TLS) Указывает файл в формате PEM со списком отзыва сертификатов.
Если эта опция используется несколько раз, применяться будет последняя.

--curves <список эллиптических кривых>

(TLS) Задаёт список эллиптических кривых, которые могут быть использованы при установлении соединения по протоколу TLS 1.3. При работе по российским криптоалгоритмам необходимо использовать кривые из списка GC256A:GC256B:GC256C:GC256D:GC512A:GC512B:GC512C.

--data-ascii <данные>

(HTTP) Это просто псевдоним для -d, --data.

--data-binary <данные>

(HTTP) Отправляет данные точно так, как указано, без какой-либо дополнительной обработки.
Если вы начинаете данные с символа @, остальное должно быть именем файла. Данные выводятся таким же образом, как и -d, --data, за исключением того, что сохраняются

новые строки и возврат каретки, а преобразования никогда не выполняются.
Как и при применении опции -d, --data тип содержимого, отправляемый на сервер, по умолчанию application/x-www-form-urlencoded. Если вы хотите, чтобы сервер обрабатывал содержимое, как произвольные двоичные данные, установите тип octeat-stream следующим образом: `-H "Content-Type: application/octet-stream"`.
Если этот параметр используется несколько раз, следующие за первым будут добавлять данные, как описано в -d, --data.

--data-raw <данные>

(HTTP) Отправляет данные аналогично -d, --data, но без специальной интерпретации символа @.
См.также -d, --data.

--data-urlencode <данные>

(HTTP) Отправляет данные, аналогичные другим параметрам -d, --data, за исключением того, что данная опция выполняет кодирование URL-адресов.
Чтобы быть совместимой с CGI, часть <данные> должна начинаться с имени, за которым следует разделитель и спецификация содержимого. Часть <данные> может быть передана в утилиту curl с помощью одного из следующих синтаксисов:

content

Указывает утилите URL-кодировать содержимое и передавать его дальше. Будьте осторожны, чтобы содержимое не содержало символов = или @, так как это приведет к тому, что синтаксис будет соответствовать одному из приведенных ниже случаев!

=content

Указывает утилите URL-кодировать содержимое и передавать его дальше. Предыдущий символ = не включен в данные.

name=content

Указывает утилите URL-кодировать часть содержимого и передавать ее дальше. Обратите внимание, что часть имени, как ожидается, уже будет закодирована в URL.

@filename

Указывает утилите загружать данные из данного файла (включая любые новые строки), закодировать в URL эти данные и передавать их POST.

name@filename

Указывает утилите загружать данные из данного файла (включая любые новые строки), закодировать в URL эти данные и передавать их POST. Часть name получает добавленный знак равенства, в результате name=urlencoded-file-content. Обратите внимание, что параметр name уже должен быть закодирован в URL.

См.также -d, --data и --data-raw.

-d, --data <данные>

(HTTP MQTT) Отправляет указанные данные в запросе POST на HTTP-сервер таким же образом, как браузер, когда пользователь заполняет HTML-форму и нажимает кнопку отправить. Это приведет к тому, что утилита передаст данные на сервер с помощью content-type application/x-www-form-urlencoded. Сравните с -F, --form.
--data-raw почти то же самое, но не имеет специальной интерпретации символа @. Чтобы отправить двоичные данные, следует использовать опцию --data-binary. Для кодирования

в URL значения поля формы можно использовать --data-urlencode.
Если какой-либо из этих параметров используется более одного раза в одной и той же командной строке, указанные фрагменты данных будут объединены вместе с разделяющим символом &. Таким образом, использование '-d name=daniel -d skill=lousy' приведет к созданию фрагмента сообщения, который выглядит как 'name=daniel&skill=lousy'.
Если в начале стоит символ '@', остальной текст должен быть именем файла для чтения данных или '-', если вы хотите, чтобы утилита считывала данные из stdin. Таким образом, отправка данных из файла с именем `"foobar"` будет выполняться с помощью -d, --data @foobar. Когда утилита с опцией -d, --data считает данные из такого файла, каретка вернется, и новые строки будут удалены. Если вы не хотите, чтобы символ @ имел специальную интерпретацию, вместо этого используйте --data-raw.
См. также --data-binary, --data-urlencode и --data-raw. Этот параметр переопределяет-F, --form и -I, --head и -T, --upload-file.

--delegation <УРОВЕНЬ>

(GSS/kerberos) Установит УРОВЕНЬ, чтобы сообщить серверу, что ему разрешено делегировать, когда дело доходит до учетных данных пользователя.

none

Не допускает никакого делегирования.

policy

Делегирует, если и только если флаг OK-AS-DELEGATE установлен в билете службы Kerberos, что является вопросом политики области.

always

Безоговорочно разрешает серверу делегировать полномочия.

--digest

(HTTP) Включает дайджест-аутентификацию HTTP(HTTP Digest Authentication). Это схема аутентификации, которая предотвращает отправку пароля в виде открытого текста. Используйте это в сочетании с обычной опцией -u, --user для установки имени пользователя и пароля.
Если эта опция используется несколько раз, применяется только первый.
См. также -u, --user, --proxy-digest и --anyauth.
Эта опция переопределяет --basic и --ntlm.

--disable-eprt

(FTP) Отключает использование команд EPRT и LPRT при выполнении активных FTP-передач. Обычно curl сначала пытается использовать EPRT, затем LPRT перед использованием команды PORT, но с этой опцией он будет использовать PORT сразу. EPRT и LPRT являются расширениями исходного протокола FTP и могут работать не на всех серверах, но они обеспечивают большую функциональность, чем традиционная команда PORT.
--eprt можно использовать для явного включения EPRT снова, а --no-eprt - это псевдоним для --disable-eprt.
Если доступ к серверу осуществляется с использованием IPv6, этот параметр не будет иметь никакого эффекта, так как в этом случае необходим EPRT.
Отключение EPRT изменяет только активное поведение. Если вы хотите переключиться в пассивный режим, вам не нужно использовать -P, --ftp-port или использовать его принудительно с --ftp-pasv.

--disable-epsv

(FTP) Отключает использование команды EPSV при выполнении пассивных FTP-передач. Обычно curl сначала пытается использовать EPSV перед PASV, но с этой опцией он не будет пытаться использовать EPSV.

--epsv можно использовать для явного включения EPSV снова, а --no-epsv-это псевдоним для --disable-epsv.

Если сервер является хостом IPv6, этот параметр не будет иметь никакого эффекта, так как в этом случае необходим EPSV.

Отключение EPSV изменяет только пассивное поведение. Если вы хотите переключиться в активный режим, вам нужно использовать -P, --ftp-port.

-q, --disable

При использовании в качестве первого параметра в командной строке конфигурационный файл curlrc не будет считываться и использоваться. См. -K, --config для получения подробной информации о пути поиска файла конфигурации по умолчанию.

--disallow-username-in-url

(HTTP) Предписывает утилите curl завершить работу, если передан URL-адрес, содержащий имя пользователя.

См. также --proto.

--dns-interface <интерфейс>

(DNS) Указывает curl отправлять исходящие DNS-запросы через <интерфейс>. Этот параметр является аналогом --interface (который не влияет на DNS). Предоставленная строка должна быть именем интерфейса (не адресом).

См. также --dns-ipv4-addr и --dns-ipv6-addr. --dns-interface требует, чтобы базовый libcurl был настроен для поддержки c-ares.

--dns-ipv4-addr <адрес>

(DNS) Указывает curl привязаться к <ip-адресу> при выполнении DNS-запросов IPv4, чтобы DNS-запросы исходили с этого адреса. Аргументом должен быть один IPv4-адрес. См. также --dns-interface и --dns-ipv6-addr. --dns-ipv4-addr требует, чтобы базовый libcurl был настроен для поддержки c-ares.

--dns-ipv6-addr <адрес>

(DNS) Указывает curl привязаться к <ip-адресу> при выполнении DNS-запросов IPv6, чтобы DNS-запросы исходили с этого адреса. Аргументом должен быть один IPv6-адрес. См.также --dns-interface и --dns-ipv4-addr. --dns-ipv6-addr требует, чтобы базовый libcurl был настроен для поддержки c-ares.

--dns-servers <адреса>

Устанавливает список DNS-серверов, которые будут использоваться вместо системного по умолчанию. Список IP-адресов должен быть разделен запятыми. Номера портов также могут быть дополнительно указаны как :<номер порта> после каждого IP-адреса.

--dns-servers требуют, чтобы базовый libcurl был настроен для поддержки c-ares.

--doh-cert-status

(все) То же самое, что и --cert-status, но используется для DOH (DNS-over-HTTPS).

--doh-insecure

(все) То же, что и -k, --insecure, но используется для DOH (DNS-over-HTTPS).

--doh-url <URL>

(все) Указывает, какой сервер DNS-over-HTTPS (DOH) следует использовать для разрешения имен хостов вместо использования механизма распознавания имен по умолчанию. URL-адрес должен быть HTTPS.

Некоторые параметры TLS, установленные для передачи, будут применяться к DOH, поскольку поиск имен выполняется по протоколу TLS. Однако параметры проверки сертификата не наследуются и могут управляться отдельно с помощью --doh-insecure и --doh-cert-status.

Если эта опция используется несколько раз, применяться будет последняя.

-D, --dump-header <имя файла>

(HTTP FTP) Пишет полученные заголовки протокола в указанный файл.

Этот параметр удобно использовать, если вы хотите сохранить заголовки, которые отправляет вам HTTP-сайт. Файлы cookie из заголовков затем могут быть прочитаны во втором вызове curl с помощью опции -b, --cookie! Опция -c, --cookie-jar - это лучший способ хранения файлов cookie.

Если заголовки не получены, использование этой опции приведет к созданию пустого файла.

При использовании в FTP, строки ответа FTP-сервера считаются `"заголовками"` и, таким образом, сохраняются там.

Если эта опция используется несколько раз, применяться будет последняя.

См. также -o, --output.

--engine <имя>

(TLS) Задаёт подгружаемый модуль engine, который предоставляет реализации алгоритмов шифрования, для работы по российским криптоалгоритмам следует использовать модуль cryptocom.

--etag-compare <файл>

(HTTP) Этот параметр создает условный HTTP-запрос для конкретного ETag, считываемого из данного файла, путем отправки пользовательского заголовка If-None-Match с использованием извлеченного ETag.

Для получения правильных результатов убедитесь, что указанный файл содержит только одну строку с требуемым ETag. Пустой файл анализируется, как пустой ETag.

Используйте опцию --etag-save, чтобы сначала сохранить ETag из ответа, а затем использовать эту опцию для сравнения с использованием сохраненного ETag в последующем запросе.

СРАВНЕНИЕ: Существует 2 типа сравнения или ETAG: Weak(слабый) и Strong(сильный). Этот параметр ожидает и использует сильное сравнение.

--etag-save <файл>

(HTTP) Этот параметр сохраняет ETag HTTP в указанный файл. Etag обычно является частью заголовков, возвращаемых запросом. Когда сервер отправляет ETag, он должен быть заключен в двойную кавычку. Этот параметр извлекает ETag без двойных кавычек и сохраняет его в <файл>.

Сервер может отправить слабый ETag, который имеет префикс `"W/"`. Этот идентификатор не учитывается, и анализируется только соответствующий ETag между кавычками.

Если ETag не был отправлен сервером или он не может быть проанализирован, создается пустой файл.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

--expect100-timeout <секунды>

(HTTP) Максимальное время в секундах, в течение которого curl ждёт ответа 100-continue, когда curl выдает заголовок Expects: 100-continue в своем запросе. По умолчанию curl будет ждать одну секунду. Эта опция принимает десятичные значения! Когда curl перестанет ждать, он продолжит работу, как если бы ответ был получен.
См.также --connect-timeout.

--fail-early

Сбой и выход при первой обнаруженной ошибке передачи.
Когда curl используется для выполнения нескольких передач в командной строке, он будет пытаться работать с каждым заданным URL-адресом один за другим. По умолчанию он будет игнорировать ошибки, если задано несколько URL-адресов, и успех последнего URL-адреса определит возвращаемый curl код ошибки. Таким образом, ранние неудачи будут `"скрыты"`последующими успешными передачами.
Используя этот параметр, curl вместо этого вернет ошибку при первом неудачном переносе, независимо от количества URL-адресов, заданных в командной строке. Таким образом, никакие сбои передачи не остаются незамеченными сценариями и тому подобными.
Этот параметр является глобальным и не должен указываться для каждого использования -:, --next.
Этот параметр не включает опцию -f, --fail, которая приводит к сбою передачи из-за кода состояния HTTP сервера. Вы можете объединить эти два варианта, однако примечание -f, --fail не является глобальным и поэтому содержится в -:, --next.

--fail-with-body

(HTTP) Возвращает ошибку при ошибках сервера, где код ответа HTTP равен 400 или больше. В обычных случаях, когда HTTP-сервер не может доставить документ, он возвращает HTML-документ, в котором говорится об этом (который часто также описывает и многое другое). Этот флаг позволит curl по-прежнему выводить и сохранять это содержимое, но также возвращать ошибку 22.
Это альтернативный вариант опции -f, --fail, которая приводит к сбою curl при тех же обстоятельствах, но без сохранения содержимого.
См. также -f, --fail.

-f, --fail

(HTTP) Сбой молча (вообще без вывода) при ошибках сервера. В основном это делается для того, чтобы скрипты и т.д. могли лучше справляться с неудачными попытками. В обычных случаях, когда HTTP-сервер не может доставить документ, он возвращает HTML-документ, в котором говорится об этом (который часто также описывает и многое другое). Этот флаг предотвратит вывод curl и вернет ошибку 22.
Этот метод не является безотказным, и бывают случаи, когда неудачные коды ответов проскальзывают, особенно когда используется аутентификация (коды ответов 401 и 407).
См. также --fail-with-body.

--false-start

(TLS) Указывает curl использовать ложный запуск во время первоначального соединения TLS (TLS handshake). Ложный запуск - это режим, в котором клиент TLS начнет отправлять данные приложения до проверки завершенного сообщения сервера, тем самым экономя время в оба конца при выполнении первоначального соединения полностью.

--form-string <имя=строка>

(HTTP SMTP IMAP) Аналогично -F, --form, за исключением того, что строка значения для именованного параметра используется буквально. Ведущие символы '@' и '<', а также строка ';type=' в аргументе, не имеют особого значения. Используйте эту опцию вместо -F, --form, если есть какая-либо вероятность того, что строковое значение может случайно вызвать функции '@' или '<' в -F, --form.

См.также -F, --form.

-F, --form <имя=содержание>

(HTTP SMTP IMAP) Позволяет curl эмулировать форму, заполненную для семейства протоколов HTTP, в которой пользователь нажал кнопку отправки. Это приводит к тому, что curl отправляет данные командой POST с использованием Content-Type multipart/form-data в соответствии с RFC 2388.

Для протоколов SMTP и IMAP это средство для создания составного почтового сообщения для передачи.

Эта опция позволяет загружать двоичные файлы и т.д. Чтобы заставить часть `"содержание"` быть файлом, нужно добавить префикс @ к имени файла. Чтобы просто получить часть содержимого из файла, префикс < к имени файла. Разница между @ и < заключается в том, что @ заставляет файл прикрепляться к сообщению в качестве загрузки файла, в то время как < создает текстовое поле и просто получает содержимое этого текстового поля из файла.

Указывает curl, чтобы он читал содержимое из stdin вместо файла, используя - как имя файла. Это относится как к конструкциям @, так и к конструкциям <. При использовании stdin содержимое сначала буферизуется в памяти curl, чтобы определить его размер и разрешить возможную повторную отправку. Определение данных детали из именованного нерегулярного файла (например, именованного канала или аналогичного), к сожалению, не подлежит буферизации и будет эффективно считываться во время передачи; поскольку полный размер неизвестен до начала передачи, такие данные отправляются в виде фрагментов по HTTP и отклоняются IMAP.

Пример: отправить изображение на HTTP-сервер, где 'profile' - это имя поля формы, к которому относится файл portrait.jpg, строка будет выглядеть так:

```
curl -F profile=@portrait.jpg https://example.com/upload.cgi
```

Пример: отправить свое имя и размер обуви в двух текстовых полях на сервер:

```
curl -F name=Alexander -F size=11 https://example.com/
```

Пример: отправить свое эссе в текстовом поле на сервер. Отправить его в виде обычного текстового поля, но получить его содержимое из локального файла:

```
curl -F "=<hugefile.txt" https://example.com/
```

Вы также можете указать curl, какой тип контента использовать, используя 'type=', например:

```
curl -F "web=@index.html;type=text/html" example.com
```

или

```
curl -F "name=daniel;type=text/foo" example.com
```

Вы также можете явно изменить имя поля части загрузки файла, установив filename=, например:

```
curl -F "file=@localfile;filename=nameinpost" example.com
```

Если имя файла/путь содержит ',' или ';', оно должно быть заключено в двойные кавычки, такие как:

```
curl -F "file=@"localfile";filename="nameinpost"" example.com
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

или

```
curl -F 'file=@"localfile";filename="nameinpost"' example.com
```

Обратите внимание, что если имя файла/путь заключено в двойные кавычки, любая двойная кавычка или обратная косая черта в имени файла должна быть экранирована обратной косой чертой.

Кавычки также должны применяться к данным, не относящимся к файлам, если они содержат точки с запятой, начальные/конечные пробелы или двойные кавычки:

```
curl -F 'colors="red; green; blue";type=text/x-myapp' example.com
```

Вы можете добавить пользовательские заголовки в поле, установив headers=, например

```
curl -F "submit=OK;headers="X-submit-type: OK"" example.com
```

или

```
curl -F "submit=OK;headers=@headerfile" example.com
```

Ключевое слово headers= может появляться несколько раз, и выше применяются примечания о цитировании. При чтении заголовков из файла пустые строки и строки, начинающиеся с '#', являются комментариями и игнорируются; каждый заголовок можно сложить, разделив его на два слова и начав строку продолжения с пробела; встроенные возвраты каретки и конечные пробелы удаляются. Вот пример содержимого файла заголовка:

```
# Этот файл содержит два заголовка.
X-Header-1: это заголовок
# Следующий заголовок сложен.
X-Header-2: это
еще один заголовок
```

Для поддержки отправки составных почтовых сообщений синтаксис расширен следующим образом:

- имя может быть опущено: знак равенства является первым символом аргумента,
- если данные начинаются с '(', это сигнализирует о начале новой составной части: за ней может следовать спецификация типа контента.
- составная часть может быть завершена аргументом'=)'.

Пример: следующая команда отправляет электронное письмо SMTP mime, состоящее из встроенной части, в двух альтернативных форматах: обычный текст и HTML. Он прикрепляет текстовый файл:

```
curl -F '=(;type=multipart/alternative' \
-F '=обычное текстовое сообщение' \
-F '= <body>HTML-сообщение</body>;type=text/html' \
-F '=)' -F '=@textfile.txt'... smtp://example.com
```

Данные могут быть закодированы для передачи с помощью encoder=. Доступные кодировки - binary и 8bit, которые не делают ничего, кроме добавления соответствующего заголовка кодирования передачи содержимого, 7bit, который отклоняет только 8-битные символы с ошибкой передачи, quoted-printable и base64, который кодирует данные в соответствии с соответствующими схемами, ограничивая длину строк до 76 символов.

Пример: отправить составную почту с текстовым сообщением для печати в кавычках и вложенным файлом base64:

```
curl -F '=текстовое сообщение;encoder=quoted-printable' \
-F '=@localfile;encoder=base64' ... smtp://example.com
```

Эту опцию можно использовать несколько раз.

Этот параметр переопределяет -d, --data и -I, --head и -T, --upload-file.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

--ftp-account <данные>

(FTP) Когда FTP-сервер запрашивает "данные учетной записи" после предоставления имени пользователя и пароля, эти данные отправляются с помощью команды ACCT. Если эта опция используется несколько раз, применяться будет последняя.

--ftp-alternative-to-user <команда>

(FTP) Если аутентификация с помощью команд USER и PASS не удалась, отправляется эта команда.

--ftp-create-dirs

(FTP) Когда FTP URL/операция использует путь, который в данный момент не существует на сервере, стандартное поведение curl - сбой. Используя эту опцию, curl вместо этого попытается создать отсутствующие каталоги.
См.также --create-dirs.

--ftp-method <метод>

(FTP) Контролирует, какой метод curl должен использоваться для доступа к файлу на FTP-сервере. Аргументом метода должен быть один из следующих вариантов:

multicwd

curl выполняет одну операцию CWD для каждой части пути в данном URL. Для глубоких иерархий это означает очень много команд. Это рекомендация RFC 1738. Это по умолчанию, но самое медленное поведение.

nocwd

curl вообще не делает CWD. curl сделает SIZE, RETR, STOR и т.д. и даёт полный путь к серверу для всех этих команд. Это самое быстрое поведение.

singlecwd

curl выполняет один CWD с полным целевым каталогом, а затем работает с файлом "нормально"(как в случае multicwd). Это несколько более соответствует стандартам, чем 'nocwd', но занимает не так много времени, как в случае 'multicwd'.

--ftp-pasv

(FTP) Используйте пассивный режим для подключения к данным. Пассивный - это внутреннее поведение по умолчанию, но с помощью этой опции можно переопределить следующую опцию -P, --ftp-port.
Если этот параметр используется несколько раз, используется только первый. Отмена принудительного пассива на самом деле невыполнима, вместо этого вы должны снова применить правильный -P, --ftp-port.
Пассивный режим означает, что curl сначала попробует команду EPSV, а затем PASV, если не используется --disable-epsv.
См. также --disable-epsv.

-P, --ftp-port <адрес>

(FTP) Отменяет роли инициатора/слушателя по умолчанию при подключении к FTP. Эта опция заставляет curl использовать активный режим. Затем curl сообщает серверу, чтобы он снова подключился к указанному адресу и порту клиента, в то время как пассивный режим просит сервер настроить IP-адрес и порт для подключения. <адрес> может быть следующим:

interface

например, "eth0 чтобы указать, какой IP-адрес интерфейса вы хотите использовать (только для Unix)

IP-address
например, «192.168.10.1» , чтобы указать точный IP-адрес

host name
например, "my.host.domain чтобы указать машину

- curl будет использовать тот же IP-адрес, который уже используется для подключения управления

Если эта опция используется несколько раз, будет использоваться последняя. Отключите использование порта с --ftp-pasv. Отключите попытку использовать команду EPRT вместо PORT с помощью --disable-eprt. EPRT-это действительно PORT++.
Вы можете добавить ":[начало]-[конец]"справа от адреса, чтобы указать curl, какой диапазон TCP-портов использовать. Это означает, что вы указываете диапазон портов от меньшего до большего числа. Можно указать один порт, но обратите внимание, что это увеличивает риск сбоя, так как порт может быть недоступен.
См. также --ftp-pasv и --disable-eprt.

--ftp-pret
(FTP) Указывает curl, отправлять команду PRET перед PASV (и EPSV). Некоторые FTP-серверы, в основном drftpd, требуют этой нестандартной команды для списков каталогов, а также для загрузки и загрузки в режиме PASV.

--ftp-skip-pasv-ip
(FTP) Указывает curl, не использовать IP-адрес, который сервер предлагает в своем ответе на команду PASV curl, когда curl подключает соединение для передачи данных. Вместо этого curl будет повторно использовать тот же IP-адрес, который он уже использует для подключения управления.
Этот параметр не действует, если вместо PASV используется порт, EPRT или EPSV.
См. также --ftp-pasv.

--ftp-ssl-ccc-mode <active/passive>
(FTP) Устанавливает режим CCC. Пассивный `(passive)` режим не инициирует завершение работы, а вместо этого ждет, пока это сделает сервер, и не будет отвечать на завершение работы с сервера. Активный `(active)` режим инициирует завершение работы и ожидает ответа от сервера.
См. также --ftp-ssl-ccc.

--ftp-ssl-ccc
(FTP) Использовать CCC (Clear Command Channel). Отключает уровень TLS после аутентификации. Остальная часть связи по каналу управления будет незашифрованной. Это позволяет маршрутизаторам NAT следить за транзакцией FTP. Режим по умолчанию-пассивный.
См.также --ssl и --ftp-ssl-ccc-mode.

--ftp-ssl-control
(FTP) Требовать TLS для входа на FTP. Обеспечивает безопасную аутентификацию, но не зашифрованную передачу данных для повышения эффективности. Если сервер не поддерживает TLS, то передача завершится с ошибкой.

-G, --get
При использовании этого параметра все данные, указанные с помощью -d, --data, --data-binary или --data-urlencode, будут использоваться в запросе HTTP GET вместо запроса

POST, который в противном случае использовался бы. Данные будут добавлены к URL-адресу с разделителем "?".
При использовании в сочетании с -I, --head данные POST вместо этого будут добавлены к URL-адресу с запросом HEAD.
Если этот параметр используется несколько раз, используется только первый. Это связано с тем, что отмена GET не имеет смысла, но вместо этого вы должны применить альтернативный метод, который вы предпочитаете.

-g, --globoff

Этот параметр отключает "анализатор глобирования URL-адресов". При установке этого параметра можно указать URL-адреса, содержащие символы [], без их интерпретации самим curl. Обратите внимание, что эти письма не являются обычным юридическим содержимым URL, но они должны быть закодированы в соответствии со стандартом URI.

--happy-eyeballs-timeout-ms <миллисекунды>

Happy eyeballs - это алгоритм, который пытается подключиться как к IPv4, так и к IPv6-адресам для хостов с двумя стеками, предпочитая сначала IPv6 в течение нескольких миллисекунд. Если IPv6-адрес не может быть подключен в течение этого времени, то параллельно предпринимается попытка подключения к IPv4-адресу. Первое соединение, которое должно быть установлено, - это то, которое используется.
Диапазон предлагаемых полезных значений ограничен. Стандарт Happy Eyeballs RFC 6555 говорит: «Рекомендуется, чтобы попытки подключения выполнялись с интервалом 150-250 мс, чтобы сбалансировать человеческий фактор и нагрузку на сеть». В настоящее время libcurl по умолчанию составляет 200 мс. Firefox и Chrome в настоящее время по умолчанию имеют значение 300 мс.
Если эта опция используется несколько раз, будет использоваться последняя.

--haproxy-protocol

(HTTP) Отправьте заголовок HAProxy PROXY protocol v1 в начале соединения. Это используется некоторыми балансировщиками нагрузки и обратными прокси-серверами для указания истинного IP-адреса и порта клиента.
Этот параметр в первую очередь полезен при отправке тестовых запросов в службу, которая ожидает этот заголовок.

-I, --head

(FILE HTTP FTP) Указывает что нужно получать только заголовки! HTTP-серверы имеют командный заголовок, который используется для получения только заголовка документа. При использовании на FTP или файловом файле curl отображает только размер файла и время последнего изменения.

-H, --header <заголовок/@файл>

(HTTP) Дополнительный заголовок для включения в запрос при отправке HTTP на сервер. Вы можете указать любое количество дополнительных заголовков. Обратите внимание, что если вы добавите пользовательский заголовок, который имеет то же имя, что и один из внутренних, которые будет использовать curl, ваш внешний заголовок будет использоваться вместо внутреннего. Это позволяет вам делать еще более сложные вещи, чем обычно делает curl. Вы не должны заменять внутренне установленные заголовки, не зная точно, что вы делаете. Удалите внутренний заголовок, указав замену без содержимого в правой части двоеточия, например:

-H "Host:". Если вы отправляете пользовательский заголовок без значения

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

curl будет следить за тем, чтобы каждый заголовок, который вы добавляете/заменяете, отправлялся с соответствующим маркером конца строки, поэтому вы не должны добавлять его как часть содержимого заголовка: не добавляйте новые строки или возврат каретки, они только испортят вам все.
Этот параметр может принимать аргумент в стиле @filename, который затем добавляет заголовок для каждой строки во входном файле. Использование @- заставит curl прочитать файл заголовка из stdin.
Используйте --proxy-header для отправки пользовательских заголовков, предназначенных для HTTP-прокси.
Передача заголовка `"Transfer-Encoding: chunked"` при выполнении HTTP-запроса с телом запроса приведет к тому, что curl отправит данные с использованием фрагментированного кодирования.

Пример:
```
curl -H "X-First-Name: Marina" http://example.com/
```

ПРЕДУПРЕЖДЕНИЕ: заголовки, установленные с помощью этой опции, будут установлены во всех запросах - даже после того, как будут выполнены перенаправления, например, при указании с помощью -L, --location. Это может привести к отправке заголовка на другие хосты, чем исходный хост, поэтому чувствительные заголовки следует использовать с осторожностью в сочетании со следующими перенаправлениями.
Этот параметр можно использовать несколько раз для добавления/замены/удаления нескольких заголовков.
См. также -A, --user-agent и -e, --referer.

-h, --help <категория>

Справка по использованию. Здесь перечислены все команды <категории>. Если аргумент не задан, curl отобразит наиболее важные аргументы командной строки. Если был указан аргумент `"all"`, curl отобразит все доступные параметры. Если был указан аргумент `"category"`, curl отобразит все категории и их значения.

--hsts <имя файла>

(HTTPS) ПРЕДУПРЕЖДЕНИЕ: этот параметр является экспериментальным. Не используйте в реальной работе.
Эта опция включает HSTS для передачи. Если имя файла указывает на существующий файл кэша HSTS, он будет использоваться. После завершения переноса кэш будет снова сохранен в имя файла, если он был изменен.
Укажите имя файла (нулевая длина), чтобы избежать загрузки/сохранения и заставить curl просто обрабатывать HST в памяти.
Если этот параметр используется несколько раз, curl загрузит содержимое из всех файлов, но последний будет использоваться для сохранения.

--http0.9

(HTTP) Сообщает, что curl будет в порядке с ответом HTTP версии 0.9.
HTTP/0.9-это полностью беззаголовочный ответ, и поэтому вы также можете подключиться к серверам, не являющимся HTTP, и все равно получить ответ, так как curl будет просто прозрачно понижен - если это разрешено.
HTTP/0.9 по умолчанию отключен.

-0, --http1.0

(HTTP) Указывает curl использовать HTTP версии 1.0 вместо использования его внутренней предпочтительной версии HTTP.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Этот параметр переопределяет --http1.1 и --http2.

--http1.1

(HTTP) Указывает curl использовать HTTP версии 1.1.
Этот параметр переопределяет -0, --http1.0 и --http2.

--http2-prior-knowledge

(HTTP) Говорит curl выдавать свои HTTP-запросы, отличные от TLS, с использованием HTTP/2 без обновления HTTP/1.1. Это требует предварительного знания того, что сервер сразу же поддерживает HTTP/2. HTTPS-запросы по-прежнему будут выполнять HTTP/2 стандартным способом с согласованной версией протокола в рукопожатии TLS.
--http2-prior-knowledge требуют, чтобы базовый libcurl был построен для поддержки HTTP/2. Этот параметр переопределяет --http1.1 и -0, --http1.0 и --http2.

--http2

(HTTP) Указывает curl использовать HTTP версии 2.
См. также --http1.1 и --http3. --http2 требует, чтобы базовый libcurl был построен для поддержки HTTP/2. Этот параметр переопределяет --http1.1 и -0, --http1.0 и --http2-prior-knowledge.

--http3

(HTTP) ПРЕДУПРЕЖДЕНИЕ: этот параметр является экспериментальным. Не используйте в коммерческом продукте.
Указывает curl использовать HTTP версии 3 непосредственно для хоста и номера порта, используемого в URL-адресе. Обычная транзакция HTTP/3 будет выполнена на хосте, а затем перенаправлена через Alt-SVc, но эта опция позволяет пользователю обойти это, когда вы знаете, что цель говорит по протоколу HTTP/3 на данном хосте и порту.
Эта опция приведет к сбою curl, если соединение QUIC не может быть установлено, оно не может самостоятельно вернуться к более низкой версии HTTP.
См. также --http1.1 и --http2. --http3 требует, чтобы базовый libcurl был построен для поддержки HTTP/3. Этот параметр переопределяет --http1.1 и -0, --http1.0 и --http2 и --http2-prior-knowledge.

--ignore-content-length

(FTP HTTP) Для HTTP игнорирует заголовок `"длина-содержимого"`. Это особенно полезно для серверов под управлением Apache 1.x, которые будут сообщать о неправильной длине содержимого для файлов размером более 2 гигабайт.
Для FTP пропустите команду RETR, чтобы определить размер перед загрузкой файла.

-i, --include

Включите заголовки ответов HTTP в выходные данные. Заголовки ответов HTTP могут включать такие вещи, как имя сервера, файлы cookie, дата документа, версия HTTP и многое другое...
Чтобы просмотреть заголовки запросов, рассмотрите опцию-v, --verbose.
См. также -v, --verbose.

-k, --insecure

(TLS) По умолчанию каждое TLS-соединение, созданное curl, проверяется на безопасность. Эта опция позволяет curl продолжать и работать даже для подключений к серверу, которые в противном случае считаются небезопасными.
Подключение к серверу проверяется, убедившись, что сертификат сервера содержит правильное имя, и успешно проверяется с помощью хранилища сертификатов.

См. Этот онлайн-ресурс для получения более подробной информации: https://curl.se/docs/sslcerts.html
См. также --proxy-insecure и --cacert.

--interface <имя>

Выполните операцию с использованием указанного интерфейса. Вы можете ввести имя интерфейса, IP-адрес или имя хоста. Пример может выглядеть следующим образом:
`curl --interface eth0:1 https://www.example.com/`
Если эта опция используется несколько раз, будет использоваться последняя.
В Linux он может использоваться для указания VRF, но двоичный файл должен либо иметь `CAP_NET_RAW`, либо запускаться от имени root.
См. также --dns-интерфейс.

-4, --ipv4

Этот параметр указывает curl разрешать имена только для адресов IPv4, а не, например, использовать IPv6.
См. также --http1.1 и --http2. Этот параметр переопределяет -6, --ipv6.

-6, --ipv6

Этот параметр указывает curl разрешать имена только для адресов IPv6, а не, например, пытаться использовать IPv4.
См. также --http1.1 и --http2. Этот параметр переопределяет -4, --ipv4.

-j, --junk-session-cookies

(HTTP) Когда curl будет предложено прочитать файлы cookie из данного файла, эта опция заставит его отказаться от всех «сеансовых файлов cookie». Это в основном будет иметь тот же эффект, что и при запуске нового сеанса. Типичные браузеры всегда отбрасывают файлы cookie сеанса, когда они закрыты.
См. также -b, --cookie и -c, --cookie-jar.

--keepalive-time <секунды>

Этот параметр задает время, в течение которого соединение должно оставаться в режиме ожидания перед отправкой запросов keepalive, и время между отдельными запросами keepalive. В настоящее время он эффективен в операционных системах, предлагающих опции сокетов `TCP_KEEPIDLE` и `TCP_KEEPINTVL` (Linux, FreeBSD). Этот параметр не действует, если используется --no-keepalive.
Если эта опция используется несколько раз, будет использоваться последняя. Если не указано, параметр по умолчанию равен 60 секундам.

--key-type <тип>

(TLS) Тип файла закрытого ключа. Укажите тип предоставленного вами закрытого ключа --key. Поддерживаются DER, PEM и ENG. Если не указано, предполагается PEM.
Если эта опция используется несколько раз, будет использоваться последняя.

--key <ключ>

(TLS) Имя файла закрытого ключа. Позволяет вам предоставить свой закрытый ключ в этом отдельном файле.
Строку, начинающуюся с PKCS11:, можно использовать для указания закрытого ключа, расположенного в устройстве PKCS#11.
Если эта опция используется несколько раз, будет использоваться последняя.

--krb <уровень>

(FTP) Включает проверку подлинности и использование Kerberos. Уровень должен быть

указан и может быть «clear», «safe», «confidential» или «private». Если вы используете уровень, который не представлен в списке, вместо этого будет использоваться «private». Если эта опция используется несколько раз, будет использоваться последняя.

--libcurl <файл>

Добавьте этот параметр в любую обычную командную строку curl, и вы получите исходный код libcurl на языке C, записанный в файл, который делает эквивалент того, что делает ваша операция командной строки!

Если этот параметр используется несколько раз, будет использоваться последнее заданное имя файла.

--limit-rate <скорость>

Указывает максимальную скорость передачи, которую вы хотите использовать curl - как для загрузки, так и для загрузки. Эта функция полезна, если у вас ограниченный канал и вы хотите, чтобы передача не использовала всю пропускную способность. Чтобы сделать это медленнее, чем было бы в противном случае.

Заданная скорость измеряется в байтах в секунду, если только не добавлен суффикс. Добавление 'k' или 'K' будет считать число в килобайтах, 'm' или 'M' – в мегабайтах, а 'g' или 'G' – в гигабайтах. Примеры: 200K, 3m и 1G.

Если вы также используете опцию -Y, --speed-limit, эта опция будет иметь приоритет и может немного повредить ограничение скорости, чтобы помочь сохранить работу логики ограничения скорости.

Если эта опция используется несколько раз, будет использоваться последняя.

-l, --list-only

(FTP POP3) (FTP) При перечислении каталога FTP этот переключатель вызывает представление только с именем. Это особенно полезно, если пользователь хочет выполнить машинный анализ содержимого каталога FTP, поскольку обычное представление каталога не использует стандартный вид или формат. При таком использовании опция вызывает отправку команды NLST на сервер вместо команды LIST

Примечание: Некоторые FTP-серверы перечисляют только файлы в своем ответе на NLST; они не включают подкаталоги и символические ссылки.

(POP3) При получении определенного сообщения электронной почты из POP3 этот переключатель заставляет выполнять команду LIST вместо RETR. Это особенно полезно, если пользователь хочет узнать, существует ли на сервере определенный идентификатор сообщения и какого он размера.

Примечание: В сочетании с -X, --request этот параметр можно использовать для отправки команды UIDL, поэтому пользователь может использовать уникальный идентификатор электронной почты, а не идентификатор сообщения для выполнения запроса.

--local-port <число/диапазон>

Устанавливает предпочтительный одиночный номер или диапазон (ОТ-ДО) номеров локальных портов, которые будут использоваться для соединения(соединений). Обратите внимание, что номера портов по своей природе являются дефицитным ресурсом, который иногда будет занят, поэтому установка этого диапазона на что-то слишком узкое может привести к ненужным сбоям в настройке соединения.

--location-trusted

(HTTP) Как -L, --location, но позволит отправить имя + пароль всем хостам, на которые сайт может перенаправить. Это может привести или не привести к нарушению безопасности, если сайт перенаправит вас на сайт, на который вы отправите свою аутентифи-

кационную информацию (которая является открытым текстом в случае базовой аутентификации HTTP).
См.также -u, --user.

-L, --location

(HTTP) Если сервер сообщает, что запрошенная страница переместилась в другое место (указано с заголовком Location: и кодом ответа 3XX), эта опция заставит curl повторить запрос на новом месте. При использовании вместе с -i, --include или -I, --head будут показаны заголовки со всех запрошенных страниц. При использовании аутентификации curl отправляет свои учетные данные только на исходный хост. Если перенаправление приведет curl к другому хосту, он не сможет перехватить пользователя+пароль. См. также --location-trusted о том, как это изменить. Вы можете ограничить количество последующих перенаправлений, используя опцию --max-redirs.
Когда curl следует за перенаправлением и если запрос является POST, он выполнит следующий запрос с GET, если ответ HTTP был 301, 302 или 303. Если код ответа был любым другим кодом 3xx, curl повторно отправит следующий запрос, используя тот же немодифицированный метод.
Вы можете сказать curl, чтобы он не изменял запросы POST, чтобы получить после 30-кратного ответа, используя для этого специальные параметры: --post301, --post302 и --post303.
Метод, заданный с помощью -X, --request, переопределяет метод, который в противном случае выбрал бы curl для использования.

--login-options <параметры>

(IMAP POP3 SMTP) Укажите параметры входа в систему, которые будут использоваться при проверке подлинности сервера.
Вы можете использовать параметры входа в систему, чтобы указать параметры протокола, которые могут использоваться во время аутентификации. В настоящее время только IMAP, POP3 и SMTP поддерживают параметры входа в систему. Для получения дополнительной информации о параметрах входа в систему см. RFC 2384, RFC 5092.
Если эта опция используется несколько раз, будет использоваться последняя.

--mail-auth <адрес>

(SMTP) Указывает единственный адрес. Это будет использоваться для указания адреса аутентификации (идентификатора) отправленного сообщения, которое ретранслируется на другой сервер.
См. также --mail-rcpt и --mail-from.

--mail-from <адрес>

(SMTP) Указывает один адрес, с которого должна отправляться данная почта.
См. также --mail-rcpt и --mail-auth.

--mail-rcpt-allowfails

(SMTP) При отправке данных нескольким получателям по умолчанию curl прерывает SMTP-диалог, если хотя бы один из получателей вызывает команду RCPT для возврата ошибки.
Поведение по умолчанию можно изменить, передав параметр командной строки --mail-rcpt-allowfails, который заставит curl игнорировать ошибки и продолжить работу с оставшимися действительными получателями.
В случае, если все получатели вызывают сбой команды RCPT TO, curl прервет SMTP-диалог и вернет ошибку, полученную от последней команды RCPT TO.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

--mail-rcpt <адрес>

(SMTP) Укажите один адрес, имя пользователя или имя списка рассылки. Повторите этот параметр несколько раз, чтобы отправить нескольким получателям.

При передаче почты получатель должен указать действительный адрес электронной почты для отправки письма.

При выполнении проверки адреса (команда VRFY) получатель должен быть указан в качестве имени пользователя или имени пользователя и домена (в соответствии с разделом 3.5 RFC 5321).

При выполнении расширения списка рассылки (команда EXPN) получатель должен быть указан с использованием имени списка рассылки, например «Friends» или «Moscow-Office».

-M, --manual

Инструкция. Печатает огромный текст справки.

--max-filesize <байт>

Указывает максимальный размер (в байтах) загружаемого файла. Если запрошенный файл больше этого значения, передача не начнется, и curl вернется с кодом выхода 63.

Можно использовать модификатор размера. Например, добавление 'k' или 'K' будет считать число в килобайтах, 'm' или 'M' – в мегабайтах, а 'g' или 'G' – в гигабайтах. Примеры: 200K, 3m и 1G.

ПРИМЕЧАНИЕ: Размер файла не всегда известен до загрузки, и для таких файлов этот параметр не действует, даже если передача файлов в конечном итоге превышает заданный предел. Это касается как FTP, так и HTTP-передач.

См. также --limit-rate.

--max-redirs <число>

(HTTP) Устанавливает максимальное количество разрешенных последовательностей перенаправления. Когда используется -L, --location, используется для предотвращения слишком частого следования curl перенаправлениям. По умолчанию ограничение установлено на 50 перенаправлений. Установите этот параметр равным -1, чтобы сделать его неограниченным.

Если эта опция используется несколько раз, будет использоваться последняя.

-m, --max-time <секунды>

Максимальное время в секундах, которое может занять вся операция. Это полезно для предотвращения зависания пакетных заданий в течение нескольких часов из-за медленных сетей или сбоев ссылок. Этот параметр принимает десятичные значения, но фактический тайм-аут будет уменьшаться по мере увеличения указанного тайм-аута в десятичной точности.

Если эта опция используется несколько раз, будет использоваться последняя.

См.также --connect-timeout.

--metalink

Этот параметр может указать curl анализировать и обрабатывать данный URI как файл Metalink (поддерживаются версии 3 и 4 (RFC 5854)) и использовать зеркала, перечисленные в нем, для отработки отказа, если есть ошибки (например, файл или сервер недоступен). Он также проверит хэш файла после завершения загрузки. Сам файл Metalink загружается и обрабатывается в памяти, а не хранится в локальной файловой системе.

Пример использования удаленного файла Metalink:

```
curl --metalink http://www.example.com/example.metalink
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Чтобы использовать файл Metalink в локальной файловой системе, используйте файловый протокол (file://):

```
curl --metalink file:///example.metalink
```

Обратите внимание, что если файловый протокол отключен, на момент написания этой статьи невозможно использовать локальный файл Metalink. Также обратите внимание, что если --metalink и -i, --include используются вместе, --include будет проигнорирован. Это связано с тем, что включение заголовков в ответ нарушит синтаксический анализатор Metalink, и если заголовки включены в файл, описанный в файле Metalink, проверка хэша завершится неудачей.

--netrc-file <имя файла>

Этот параметр аналогичен -n, --netrc, за исключением того, что вы указываете путь (абсолютный или относительный) к файлу netrc, который должен использовать curl. Вы можете указать только один файл netrc для каждого вызова. Если предусмотрено несколько параметров --netrc-файла, будет использоваться последний.
Он будет соблюдать --netrc-optional, если указано.
Этот параметр переопределяет -n, --netrc.

--netrc-optional

Очень похоже на -n, --netrc, но эта опция делает использование .netrc необязательным и не обязательным, как опция -n, --netrc.
См. также --netrc-файл. Этот параметр переопределяет -n, --netrc.

-n, --netrc

Заставляет curl сканировать файл .netrc (_netrc в Windows) в домашнем каталоге пользователя на предмет имени пользователя и пароля. Обычно это используется для FTP в Unix. При использовании с HTTP curl включит аутентификацию пользователя. Curl не будет жаловаться, если этот файл не имеет правильных разрешений (он не должен быть доступен для чтения всем или группе). Переменная среды "HOME"используется для поиска домашнего каталога.

```
Быстрый и очень простой пример того, как настроить .netrc, чтобы разрешит

machine host.domain.com login myself password secret
```

-:, --next

Указывает curl использовать отдельную операцию для следующего URL-адреса и связанных с ним параметров. Это позволяет отправлять несколько запросов URL-адресов, каждый из которых имеет свои собственные параметры, например, различные имена пользователей или пользовательские запросы для каждого из них.
-:, --next сбросит все локальные параметры, и только глобальные сохранят свои значения для операции, следующей за инструкцией -:, --next. Глобальные параметры включают -v, --verbose, --trace, --trace-ascii и --fail-early.
Например, вы можете выполнить как GET, так и POST в одной командной строке:

```
curl www1.example.com --next -d postthis www2.example.com
```

--no-alpn

(HTTPS) Отключает расширение ALPN TLS. ALPN по умолчанию включёи и используется libcurl, поддерживающим HTTP/2, для согласования поддержки HTTP/2 с сервером во время сеансов https.
См. также --no-npn и --http2. --no-alpn.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

-N, --no-buffer

Отключает буферизацию выходного потока. В обычных рабочих ситуациях curl будет использовать стандартный буферизованный выходной поток, который приведет к тому, что он будет выводить данные по частям, не обязательно точно, когда данные поступят. Использование этой опции отключит эту буферизацию.

Обратите внимание, что это задокументированное имя отрицаемого параметра. Таким образом, вы можете использовать --buffer для обеспечения буферизации.

--no-keepalive

Отключает использование сообщений keepalive в TCP-соединении. в противном случае curl включает их по умолчанию.

Обратите внимание, что это задокументированное имя отрицаемого параметра. Таким образом, вы можете использовать --keepalive для принудительного выполнения keepalive.

--no-npn

(HTTPS) Отключает расширение NPN TLS. NPN включен по умолчанию и используется libcurl, поддерживающим HTTP/2, для согласования поддержки HTTP/2 с сервером во время сеансов https.

См. также --no-alpn и --http2. --no-npn.

--no-progress-meter

Возможность отключения вывода индикатора прогресса без приглушения или иного воздействия на предупреждающие и информационные сообщения, такие как -s, --silent.

Обратите внимание, что это задокументированное имя отрицаемого параметра. Таким образом, вы можете использовать --progress-meter, чтобы снова включить индикатор прогресса.

См. также -v, --verbose и -s, --silent.

--no-sessionid

(TLS) Отключает использование curl кэширования идентификаторов сеансов TLS. По умолчанию все передачи выполняются с использованием кэша. Обратите внимание, что, хотя ничто не должно пострадать при попытке повторного использования идентификаторов сеансов TLS, в интернете, похоже, существуют дефектные реализации TLS, которым может потребоваться отключение этого для их работы.

Обратите внимание, что это задокументированное имя отрицаемого параметра. Таким образом, вы можете использовать --sessionid для принудительного кэширования идентификатора сеанса.

`Разделенный запятыми список хостов, которые не используют прокси-сервер,`

Этот параметр переопределяет переменные среды, отключающие прокси-сервер. Если есть переменная среды, отключающая прокси-сервер, вы можете установить для списка noproxy значение , чтобы переопределить его.

--ntlm-wb

(HTTP) Включает NTLM, практически так же как опция --ntlm, но передает аутентификацию отдельному двоичному приложению ntlmauth, которое выполняется при необходимости.

См. также --ntlm и --proxy-ntlm.

--ntlm

(HTTP) Включает проверку подлинности NTLM. Метод проверки подлинности NTLM был разработан корпорацией Майкрософт и используется веб-серверами IIS. Это про-

приетарный протокол, полученный методом обратной разработки соответствующими специалистами и реализованный в curl на основе их усилий. Такое поведение не следует одобрять, вы должны побуждать всех, кто использует NTLM, переключаться на общедоступный и документированный метод аутентификации, такой как Digest.
Если вы хотите включить NTLM для проверки подлинности прокси-сервера, используйте --proxy-ntlm.
Если этот параметр используется несколько раз, используется только первый.
См. также --proxy-ntlm. --ntlm.
Этот параметр переопределяет --basic, --digest и --anyauth.

--oauth2-bearer <токен>

(IMAP POP3 SMTP HTTP) Указывает токен носителя для проверки подлинности сервера OAUTH 2.0. Токен на предъявителя используется в сочетании с именем пользователя, которое может быть указано как часть параметров --url или -u, --user.
Токен на предъявителя и имя пользователя отформатированы в соответствии с RFC 6750.
Если эта опция используется несколько раз, будет использоваться последняя.

--output-dir <dir>

Этот параметр указывает каталог, в котором должны храниться файлы, когда используются -O, --remote-name или -o, --output.
Данный выходной каталог используется для всех URL-адресов и параметров вывода в командной строке, вплоть до первого -:, --next.
Если указанный целевой каталог не существует, операция завершится неудачно, если не будет также использоваться --create-dirs.
Если этот параметр используется несколько раз, будет использоваться последний указанный каталог.
См. Также -O, --remote-name и -J, --remote-header-name.

-o, --output <файл>

Перенаправляет выходные данные в <файл> вместо stdout. Если вы используете или [] для извлечения нескольких документов, вы должны указать URL-адрес, и вы можете использовать "# за которым следует число в спецификаторе <file>. Эта переменная будет заменена текущей строкой для извлекаемого URL-адреса. Как в:

```
curl "http://{один,два}.example.com" -o "file_#1.txt"
```

или используйте несколько переменных, таких как:

```
curl "http://{site,host}.host[1-5].com" -o "#1_#2"
```

Вы можете использовать эту опцию столько раз, сколько у вас есть URL-адресов. Например, если вы укажете два URL-адреса в одной командной строке, вы можете использовать его следующим образом:

```
curl -o aa example.com -o bb example.net
```

и порядок параметров -o и URL-адресов не имеет значения, просто первый -o предназначен для первого URL-адреса и так далее, поэтому приведенную выше командную строку также можно записать как

```
curl example.com example.net -o aa -o вв
```

См. также параметр --create-dirs для динамического создания локальных каталогов. Указание вывода как "(один дефис) приведет к тому, что вывод будет выполнен в stdout.
См. Также -O, --remote-name, --remote-name-all и -J, --remote-header-name.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

--parallel-immediate

При выполнении параллельных передач эта опция проинструктирует curl, что он должен предпочитать открывать несколько соединений параллельно одновременно, а не ждать, чтобы увидеть, можно ли добавить новые передачи в виде мультиплексированных потоков в другом соединении.

См. также -Z, --parallel и --parallel-max.

--parallel-max

При выполнении параллельных передач с использованием опции -Z, --parallel, этот параметр управляет максимальным количеством передач, выполняемых одновременно.

Значение по умолчанию равно 50.

См. также -Z, --parallel.

-Z, --parallel

Указывает curl выполнять передачи параллельно, а не последовательно.

--pass <фраза>

(SSH TLS) Парольная фраза для закрытого ключа

Если эта опция используется несколько раз, будет использоваться последняя.

--path-as-is

Указывает curl, чтобы он не обрабатывал последовательности /../ или /./ в заданном URL-пути. Обычно curl редуцирует или объединит их в соответствии со стандартами, но эта опция отменяет это действие.

--pinnedpubkey <хэши>

(TLS) Указывает curl использовать указанный файл открытого ключа (или хэши) для проверки однорангового узла. Это может быть путь к файлу, содержащему один открытый ключ в формате PEM или DER, или любое количество хэшей sha256 в кодировке base64, которым предшествует sha256// и которые разделены ;

При согласовании соединения TLS сервер отправляет сертификат, указывающий его личность. Открытый ключ извлекается из этого сертификата, и если он не точно соответствует открытому ключу, предоставленному для этой опции, curl прервет соединение перед отправкой или получением каких-либо данных.

Если эта опция используется несколько раз, будет использоваться последняя.

--post301

(HTTP) Указание работать по стандарту 7231/6.4.2 и не преобразовывать запросы POST в запросы GET при перенаправлении 301. Поведение, отличное от RFC, повсеместно встречается в веб-браузерах, поэтому curl выполняет преобразование по умолчанию для поддержания согласованности. Однако сервер может потребовать, чтобы запрос POST оставался POST после такого перенаправления. Этот параметр имеет смысл только при использовании -L, --location.

См.также --post302, --post303 и -L, --location.

--post302

(HTTP) Указание работать по стандарту RFC 7231/6.4.3 и не преобразовывать запросы POST в запросы GET при перенаправлении 302. Поведение, отличное от RFC, повсеместно встречается в веб-браузерах, поэтому curl выполняет преобразование по умолчанию для поддержания согласованности. Однако сервер может потребовать, чтобы запрос POST оставался POST после такого перенаправления. Этот параметр имеет смысл только при использовании -L, --location.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

См.также --post301, --post303 и -L, --location.

--post303

(HTTP) Указание нарушать стандарт RFC 7231/6.4.4 и не преобразовывать запросы POST в запросы GET при следующих 303 перенаправлениях. Сервер может потребовать, чтобы запрос POST оставался POST после перенаправления 303. Этот параметр имеет смысл только при использовании -L, --location.
См.также --post302, --post301 и -L, --location.

--preproxy [протокол://]хост[:порт]

Использовать указанный прокси-сервер SOCKS перед подключением к HTTP или HTTPS -x, --proxy. В таком случае curl сначала подключается к прокси-серверу SOCKS, а затем подключается (через SOCKS) к прокси-серверу HTTP или HTTPS. Таков механизм препрокси.
`Строка предварительного прокси-сервера должна быть указана с префиксом pr`
`Если номер порта не указан в строке прокси-сервера, предполагается, что с`

`Пользователь и пароль, которые могут быть указаны в строке прокси-сервера`
Если эта опция используется несколько раз, будет использоваться последняя.

-#, --progress-bar

Отображать прогресс передачи в виде простого индикатора выполнения вместо стандартного, более информативного счетчика.
Этот индикатор выполнения рисует одну строку символов "#"по экрану и показывает процент, если известен размер передачи. Для передач без известного размера будут показаны символы (-=o=-), который перемещается вперед и назад, но только во время передачи данных, с набором летающих символов хэш-знака сверху.

--proto-default <протокол>

Указывает curl использовать <протокол> для любого URL-адреса, в котором отсутствует имя схемы.
Пример:

curl --proto-default https ftp.mozilla.org

Неизвестный или неподдерживаемый протокол вызывает ошибку `CURLE_UNSUPPORTED_PROTOCOL`.
Этот параметр не изменяет протокол прокси-сервера по умолчанию (http).
Без этой опции curl сделал бы предположение, основанное на хосте, см. --url для получения подробной информации.

--proto-redir <протоколы>

Указывает curl ограничить, какие протоколы он может использовать при перенаправлении. Протоколы, запрещенные --proto, не переопределяются этой опцией. См. --proto для представления протоколов.
Например, разрешить только HTTP и HTTPS при перенаправлении:
`curl --proto-redir -all,http,https http://example.com`
По умолчанию curl разрешает HTTP, HTTPS и FTP при перенаправлении. Указание all или +all включает все протоколы перенаправления, включая те, которые отключены для обеспечения безопасности.

--proto <протоколы>

Указывает curl ограничить, какие протоколы он может использовать при передаче. Про-

токолы оцениваются слева направо, разделяются запятыми и каждый из них является именем протокола или "all необязательно с префиксом ноль или более модификаторов. Доступны следующие модификаторы:
+ Разрешить этот протокол в дополнение к уже разрешенным протоколам (это значение по умолчанию, если модификатор не используется).
– Отклонить этот протокол, удалив его из списка уже разрешенных протоколов.
= Разрешить только этот протокол (игнорируя уже разрешенный список), хотя он может быть изменен последующими записями в списке, разделенном запятыми.
Например:
--proto -ftps использует протоколы по умолчанию, но отключает ftps
--proto -all,https,+http включает только http и https
--proto =http,https также включает только http и https
Неизвестные протоколы выдают предупреждение. Это позволяет сценариям безопасно полагаться на возможность отключения потенциально опасных протоколов, не полагаясь на поддержку этого протокола, встроенного в curl, чтобы избежать ошибки.
Этот параметр можно использовать несколько раз, и в этом случае эффект будет таким же, как при объединении протоколов в один экземпляр параметра.
См. также --proto-redir и --proto-default.

--proxy-anyauth
Указывает curl выбрать подходящий метод аутентификации при общении с данным HTTP-прокси. Это может привести к дополнительному запросу/ответу.
См. также -x, --proxy, --proxy-basic и --proxy-digest.

--proxy-basic
Указывает curl использовать базовую аутентификацию HTTP при общении с данным прокси-сервером. Используйте --basic для включения HTTP Basic с удаленным хостом. Basic - это метод аутентификации по умолчанию, который curl использует с прокси-серверами.
См. также -x, --proxy, --proxy-anyauth и --proxy-digest.

--proxy-cacert <файл>
То же самое, что и --cacert, но используется для прокси-сервера HTTPS.
См. также --proxy-capath, --cacert, --capath и -x, --proxy.

--proxy-capath <папка>
То же самое, что и --capath, но используется для прокси-сервера HTTPS.
См. также --proxy-cacert, -x, --proxy и --capath.

--proxy-cert-type <тип>
То же самое, что и --cert-type, но используется для прокси-сервера HTTPS.

--proxy-cert <сертификат[:пароль]>
То же самое, что и -E, --cert, но используется для прокси-сервера HTTPS.

--proxy-ciphers <list>
То же самое, что и --ciphers, но используется для прокси-сервера HTTPS.

--proxy-crlfile <файл>
То же самое, что и --crlfile, но используется для прокси-сервера HTTPS.

--proxy-digest
Указывает curl использовать дайджест-аутентификацию HTTP при общении с указанным

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

прокси-сервером. Используйте --digest для включения HTTP-дайджеста с удаленным хостом.
См. также -x, --proxy, --proxy-anyauth и --proxy-basic.

--proxy-header <заголовок/@файл>
(HTTP) Дополнительный заголовок для включения в HTTP-запрос при отправке на прокси-сервер. Вы можете указать любое количество дополнительных заголовков. Это аналог опции -H, --header, но предназначенный только для прокси-соединения в запросах CONNECT, когда требуется чтобы на прокси-сервер отправлялся отдельный заголовок, отличный от того что отправляется на удаленный хост.
curl следит за тем, чтобы каждый заголовок, который вы добавляете/заменяете, отправлялся с соответствующим маркером конца строки: не добавляйте вручную символы новой строки и возврата каретки.
Заголовки, указанные с помощью этого параметра, не будут включатся в запросы, которые не будут отправлены прокси-серверу.
Этот параметр может принимать аргументы в стиле @имя-файла, каждая строка которого будет добавлена как заголовок. При указании аргумента @- curl будет читать файл заголовка из stdin.
Этот параметр можно использовать несколько раз для добавления/замены/удаления нескольких заголовков.

--proxy-insecure
То же, что и -k, --insecure, но используется для прокси-сервера HTTPS.

--proxy-key-type <тип>
То же самое, что и --key-type, но используется для прокси-сервера HTTPS.

--proxy-key <ключ>
То же самое, что и --key, но используется для прокси-сервера HTTPS.

--proxy-ntlm
Указывает, что curl должен использовать аутентификацию HTTP NTLM при соединении с прокси-сервером. Укажите --ntlm для включения аутентификации NTLM при соединении с удаленным хостом.
См. также --proxy-anyauth.

--proxy-pass <пароль>
То же самое, что и --pass, но используется для прокси-сервера HTTPS.

--proxy-pinnedpubkey <хэши>
(TLS) Указывает, что curl должен использовать указанный файл открытого ключа (или хэши) для проверки прокси-сервера. Это может быть путь к файлу, содержащему один открытый ключ в формате PEM или DER, или любое количество хэшей sha256 в кодировке base64, которым предшествует `'sha256//' и которые разделены ';'.`
При установке соединения TLS сервер отправляет сертификат для идентификации. Открытый ключ извлекается из этого сертификата, и если он не соответствует открытому ключу, указанному как аргумент этой опции, curl прервет соединение, не отправляя и не получая никаких данных.
Если эта опция указана несколько раз, будет использоваться последняя.

--proxy-service-name <имя>
Этот параметр позволяет изменить имя службы для соединения с прокси-сервером.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

--proxy-ssl-allow-beast
То же самое, что и --ssl-allow-beast, но используется для прокси-сервера HTTPS.

--proxy-ssl-auto-client-cert
То же самое, что и --ssl-auto-client-cert, но используется для прокси-сервера HTTPS.

--proxy-tls13-ciphers <список шифров>
(TLS) Указывает, какие наборы шифров следует использовать при подключении к прокси-серверу HTTPS при соединении TLS 1.3. В качестве аргументов должны быть указаны допустимые шифры.
Если эта опция указана несколько раз, будет использоваться последняя.

--proxy-tlsauthtype <тип>
То же самое, что и --tlsauthtype, но используется для прокси-сервера HTTPS.

--proxy-tlspassword <строка>
То же самое, что и --tlspassword, но используется для прокси-сервера HTTPS.

--proxy-tlsuser <имя>
То же самое, что и --tlsuser, но используется для прокси-сервера HTTPS.

--proxy-tlsv1
То же самое, что и -1, --tlsv1, но используется для прокси-сервера HTTPS.

-U, --proxy-user <пользователь:пароль>
Задает имя пользователя и пароль, которые будут использоваться для проверки подлинности прокси-сервера.
В ОС Windows, если вы указали Negotiate или проверку подлинности NTLM, то curl может выбрать имя пользователя и пароль из вашей среды, указав двоеточие в качестве аргумента: `"-U :"`.
В операционных системах, где это возможно, curl скроет данный аргумент опции из списков процессов. Этого недостаточно, чтобы защитить учетные данные от возможного просмотра другими пользователями в той же системе, поскольку они все равно будут видны в течение короткого времени, прежде чем будут очищены. Для безопасности имя пользователя и пароль рекомендуется считывать из файла, а не указывать в качестве опций командной строки.
Если эта опция указана несколько раз, будет использоваться последняя.

-x, --proxy `[протокол://]хост[:порт]`
Использовать прокси-сервер с указанными параметрами.
Строка прокси-сервера может быть указана с префиксом `протокол://`. Если указан `http://` или протокол не указан то будет считаться, что установлен HTTP-прокси. Используйте `socks4://`, `socks4a://`, `socks5://` или `socks5h://`, чтобы запросить конкретную версию SOCKS для использования.
Если указан `https://`, то будет использоваться протокол HTTPS.
Если номер порта не указан в строке прокси-сервера, предполагается, что он равен 1080.
Этот параметр переопределяет существующие переменные среды, которые задают использование прокси-сервера. Если есть переменная среды, устанавливающая прокси-сервер, вы можете установить прокси как пустое значение `''`, чтобы переопределить её.
Все операции, выполняемые через HTTP-прокси, будут прозрачно преобразованы в HTTP. Это означает, что определенные операции, специфичные для протокола, могут быть недоступны. Это не относится к случаю использования прокси как туннеля, например, через задание опции -p, --proxytunnel.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Имя пользователя и пароль необходимо указывать в формате URL. Это позволяет передавать специальные символы, такие как `@`, указав код `%40` или передавать двоеточие кодом `%3a`.
Хост прокси-сервера можно указать точно так же, как и переменные среды прокси-сервера, включая префикс протокола `(http://)` а также пользователя прокси и пароль. Если эта опция указана несколько раз, будет использоваться последняя.

--proxy1.0 <хост[:порт]>

Используйте указанный прокси-сервер HTTP 1.0. Если номер порта не указан, предполагается, что он находится в порту 1080.
Единственное различие между этим и опцией HTTP proxy-x, --proxy, заключается в том, что при попытке использовать CONNECT через прокси-сервер будет указан протокол HTTP 1.0 вместо HTTP 1.1 по умолчанию.

-p, --proxytunnel

Когда используется HTTP-прокси (-x, --proxy), эта опция устанавливает туннель через прокси-сервер. Туннеллирование выполняется с помощью запроса HTTP-прокси CONNECT и требует, чтобы прокси-сервер разрешал прямое подключение к удаленному порту, через который curl будет устанавливать туннель.
Чтобы не выводить заголовки ответов CONNECT, когда curl настроен на вывод заголовков, укажите опцию --suppress-connect-headers.
См. также -x, --proxy.

-Q, --цитата

(FTP) Отправляет произвольную команду на удаленный сервер FTP. Команды Quote отправляются ДО начала передачи (точнее - сразу после начальной команды PWD при передаче по FTP). Чтобы команды выполнялись после успешной передачи, поставьте перед ними тире `"-"`. Чтобы команды отправлялись после того, как curl изменил рабочий каталог, непосредственно перед командой (командами) передачи укажите `"+"` (поддерживается только для FTP). Вы можете указать любое количество команд.
Если сервер вернет ошибку для одной из команд, все операции будут прерваны. Поддерживаются команды FTP указанные в стандарте RFC 959 для отправки на FTP-серверы.
Укажите символ звездочки `('*')` перед командой, чтобы curl продолжал работать, даже если команда завершится неудачно (по умолчанию curl остановится при первой ошибке).
Эту опцию можно указывать несколько раз.

-r, --range <диапазон>

(HTTP FTP файл) Извлекает последовательность байт (т. е. часть документа) с сервера HTTP/1.1, FTP или локального файла. Диапазоны могут быть заданы несколькими способами.
0-499 задает первые 500 байт

500-999 задает вторые 500 байт

-500 задает последние 500 байт

9500- задает байты, начиная с 9500 и далее

0-0,-1 задает только первый и последний байт(*)(HTTP)

100-199,500-599 задает два отдельных 100-байтовых диапазона(*) (HTTP)
Обратите внимание, что это приведет к тому, что сервер ответит составным ответом, который curl вернет `"как есть"`! Анализ или иное преобразование этого ответа является

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

обязанностью составителя запроса.
В качестве диапазона допустимо указывать только цифры (0-9). Если в диапазоне указан нецифровой символ, ответ сервера будет зависеть от конфигурации сервера.
Учтите, что многие сервера HTTP/1.1 не поддерживают эту функцию, поэтому при попытке получить диапазон вы получите весь документ.
Загрузка диапазона по FTP поддерживает только простой синтаксис `"начало-конец"` (одно из чисел можно опустить). Работа по протоколу FTP зависит от расширенной команды FTP SIZE.
Если эта опция указана несколько раз, будет использоваться последняя.

--raw

(HTTP) Если указано, то вся внутренняя обработка и перекодирование HTTP-содержимого отключается и содержимое передается неизменным.

-e, --referer <URL>

(HTTP) Отправляет страницу `"Referer"` на HTTP-сервер. Такое же поведение можно также задать с помощью флага -H, --header. При использовании с -L, --location вы можете добавить `";auto"` к URL-адресу, указанному опцией -e, --referer, чтобы curl автоматически задавал предыдущий URL-адрес, когда он следует за заголовком Location. Аргумент `";auto"` можно использовать даже если вы не указали -e, --referer.
Если эта опция указана несколько раз, будет использоваться последняя.
См. также -A, --user-agent и -H, --header.

-J, --remote-header-name

(HTTP) Этот параметр задает, что опция -O, --remote-name должна использовать указанное сервером в Content-Disposition имя файла, вместо использования имени файла из URL-адреса.
Если сервер указывает имя файла и файл с этим именем уже существует в текущем рабочем каталоге, он не будет перезаписан и возникнет ошибка. Если сервер не указывает имя файла, то этот параметр не имеет никакого эффекта.
Перекодировка `%-символов` в имени файлов не выполняется, поэтому выходной файл может быть не тем, который вы ожидаете.
ПРЕДУПРЕЖДЕНИЕ: Будьте осторожны используя эту опцию, особенно в ОС Windows. Сервер злоумышленника может отправить вам имя DLL или другого файла, который может быть автоматически загружен ОС Windows или каким-либо сторонним программным обеспечением.

--remote-name-all

Эта опция изменяет действие по умолчанию для всех заданных URL-адресов, которые будут обрабатываться так, как если бы для каждого из них указана опция '-O, --remote-name'. Чтобы отключить это поведение для определенного URL-адреса после использования --remote-name-all, вы должны использовать -o - или --no-remote-name.

-O, --remote-name

Сохраняет вывод в локальный файл с таким же именем, которое имеет удаленный файл, который был загружен (сохраняется непосредственно имя файла, полный путь отбрасывается).
Файл будет сохранен в текущем рабочем каталоге. Если вы хотите, чтобы файл был сохранен в другом каталоге, убедитесь, что вы изменили текущий рабочий каталог, прежде чем вызывать curl с этой опцией.
Имя удаленного файла, которое будет использоваться для сохранения, извлекается из

заданного URL-адреса, если локальный файл уже существует, то он будет перезаписан. Если вы хотите, чтобы сервер мог выбрать имя файла, укажите опцию '-J, --remote-header-name', которую можно использовать в дополнение к этой опции. Если сервер выберет имя файла, и этот файл уже существует, то он не будет перезаписан.
В имени файла не выполняется декодирование URL-адреса. Если в имени присутствует символ '
Вы можете указать эту опцию для каждого URL-адреса.

-R, --remote-time

При задании этой опции curl постарается получить дату и время удаленного файла и установит локальному файлу такие же дату и время.

--request-target

(HTTP) Указывает curl использовать альтернативную `"цель"` (путь) вместо использования пути, указанного в URL-адресе. Особенно полезно, когда требуется выдавать HTTP-запросы без ведущей косой черты или других данных, которые не следуют обычному шаблону URL, например `"OPTIONS *"`.

-X, --request <команда>

(HTTP) Указывает пользовательский метод запроса, используемый при взаимодействии с HTTP-сервером. Указанный метод запроса будет использоваться вместо метода, используемого по умолчанию (по умолчанию используется метод GET). Подробности и пояснения см. в спецификации HTTP 1.1. Распространенные HTTP-запросы включают в себя PUT и DELETE, но некоторые протоколы, такие как WebDAV, предлагают запросы PROPFIND, COPY, MOVE и многие другое.
Обычно указание этой опции не требуется. Все виды запросов GET, HEAD, POST и PUT чаще вызываются с помощью отдельных параметров командной строки.
Этот параметр изменяет только фактическое слово, используемое в HTTP-запросе, он не изменяет поведение curl. Так, например, если вы хотите сделать правильный запрос HEAD, использования -X HEAD будет недостаточно. Вам нужно использовать опцию -I, --head.
Строка метода, заданная с помощью -X, --request, будет использоваться для всех запросов. Например, при использовании -L, --location, могут появиться непреднамеренные побочные эффекты, так как curl не изменяет метод запроса в соответствии с кодами ответов HTTP 30x.
(FTP) Указывает пользовательскую команду FTP для использования вместо LIST при создании списков файлов с помощью FTP.
(POP3) Указывает пользовательскую команду POP3 для использования вместо LIST или RETR.
(IMAP) Указывает пользовательскую команду IMAP для использования вместо LIST.
(SMTP) Указывает пользовательскую команду SMTP для использования вместо HELP или VRFY.
Если эта опция указана несколько раз, будет использоваться последняя.

--resolve <[+]хост:порт:адрес[,адрес]...>

Задает пользовательский адрес для конкретной пары хостов и портов. Используя это, вы можете заставить запросы curl использовать указанный адрес и запретить использование адреса, используемого обычно. Эта опция может рассматриваться как альтернатива файлу /etc/hosts, только указываемая в командной строке. Номер порта должен быть номером, используемым для конкретного протокола, для которого будет использоваться

хост. Это означает, что вам нужно несколько записей, если вы хотите указать адрес для одного и того же хоста, но разных портов.
Указав "*" в качестве хоста, вы можете указать curl разрешить любую пару хостов и конкретных портов по указанному адресу. Такие записи используются последними, поэтому любое --resolve с определенным хостом и портом будет использоваться первым. Указанный адрес, установленный этой опцией, будет использоваться, даже если -4, --ipv4 или -6, --ipv6 предписывает использование curl другой версии протокола IP.
Поставив перед хостом префикс "+", вы можете сделать тайм-аут входа после тайм-аута по умолчанию curl (1 минута). Обратите внимание, что это будет иметь смысл только для длительных параллельных передач с большим количеством файлов. В таких случаях, если используется этот параметр, curl попытается разрешить хост, как обычно, после истечения тайм-аута.
Эту опцию можно указывать много раз, чтобы добавить много имен хостов для разрешения.

--retry-all-errors

Повторять попытку при любой ошибке. Этот параметр используется вместе с --retry.
Не используйте эту опцию по умолчанию, могут возникнуть непредвиденные последствия, такие как отправка или получение дубликатов данных. Не используйте с перенаправленным входом или выходом. Пожалуйста, прочитайте пример ниже.
Предупреждение: Для совместимости с сервером curl пытается повторить неудачную передачу данных как можно ближе к тому, как они были отправлены, но это невозможно при перенаправленном вводе или выводе. Например, перед повторной попыткой он удаляет выходные данные из неудачной частичной передачи, которая была записана в выходной файл. Однако это не относится к данным, перенаправленным в файл через «конвейер» (|) или через ">", которые не сбрасываются. Настоятельно рекомендуется не анализировать и не записывать выходные данные через перенаправление в сочетании с этой опцией, так как вы можете получить дубликаты данных.
По умолчанию curl не будет выдавать ошибку при ответе HTTP, если передача прошла успешно. Например, если сервер возвращает ошибку 404 и ответ полностью получен, то это не считается ошибкой. Когда используется --retry, curl повторит попытку для некоторых кодов ответов HTTP, которые указывают на временные ошибки HTTP, но это не включает большинство кодов ответов 4xx, таких как 404. Если вы хотите повторить отправку для всех кодов ответов, указывающих на ошибки HTTP (4xx и 5xx), то в дополнение к этой опции укажите -f, --fail.

--retry-connrefused

В дополнение к другим условиям, рассматривает ошибку ECONNREFUSED как временную ошибку для опции --retry. Этот параметр используется вместе с --retry.

--retry-delay <секунды>

Если задано, то curl будет ожидать указанное количество времени перед каждой повторной попыткой, когда передача завершилась неудачно с временной ошибкой (опция заменяет задержку по умолчанию). Эта опция имеет значение только в том случае, если указана опция --retry. Установка этой задержки на ноль заставит curl использовать задержку по умолчанию.
Если эта опция указана несколько раз, будет использоваться последняя.

--retry-max-time <секунды>

Таймер повторных запросов сбрасывается перед первой попыткой передачи. Повторные

попытки (см. --retry) будут выполняться до тех пор, пока значение не достигнет предела, заданного этой опцией. Обратите внимание, что если таймер не достиг предела, то запрос будет отправлен и может выполняться дольше, чем данный период времени. Чтобы ограничить максимальное время одного запроса, используйте опцию -m, --max-time. Установите этот параметр равным нулю, чтобы не повторять запросы.
Если эта опция указана несколько раз, будет использоваться последняя.

--retry <кол-во>

Если во время передачи curl получает временную ошибку, то передача будет повторяться указанное количество раз, прежде чем завершиться с ошибкой. Установка числа в 0 указывает, что curl не должен делать повторных передач (по умолчанию). Временная ошибка означает следующее: время ожидания истекло, получен код ответа FTP 4xx, либо код ответа HTTP 408, 429, 500, 502, 503 или 504.
При повторных передачах curl будет ожидать 1 секунду, а затем для всех предстоящих повторных попыток удвоит время ожидания, пока оно не достигнет 10 минут, что затем будет задержкой между остальными повторными попытками. Указав опцию --retry-delay, этот алгоритм экспоненциального увеличения задержки будет отключен. См. также опцию --retry-max-time, для ограничения ограничить общего интервала, разрешенного для повторных попыток.
При получении в ответе заголовка Retry-After:, если такой заголовок получен, curl будет использовать это значение для следующей попытки передачи.
Если эта опция указана несколько раз, будет использоваться последняя.

--sasl-authzid <идентификатор>

Задает идентификатор авторизации (authzid) во время аутентификации SASL PLAIN в дополнение к идентификатору аутентификации (authcid), указанному в опции -u, --user. Если эта опция не указана то сервер будет получать идентификатор authzid из идентификатора authcid, но если опция указана, то, в зависимости от реализации сервера, идентификатор может использоваться для доступа к почтовому ящику другого пользователя, если к нему предоставлен доступ, или, например, к общему почтовому ящику.

--sasl-ir

Включает начальный ответ при проверке подлинности SASL.

-S, --show-error

Если задана опция -s, --silent, curl будет показывать сообщение об ошибке, если она возникает.
См. также --no-progress-meter.

-s, --silent

Бесшумный или тихий режим. Индикатор выполнения или сообщения об ошибках не отображается. Curl не выводит никаких сообщений. Curl по-прежнему будет выводить данные, которые вы запрашиваете, даже на экран консоли/stdout, если вы не перенаправите вывод.
Используйте -S, --show-error в дополнение к этой опции, чтобы отключить индикатор выполнения, но по-прежнему показывать сообщения об ошибках.
См. также -v, --verbose, --stderr и --no-progress-meter.

--socks4 <хост[:порт]>

Задает указанный прокси-сервер SOCKS4. Если номер порта не указан, предполагается, что он находится в порту 1080.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Этот параметр переопределяет любое предыдущее использование -x, --proxy, поскольку они являются взаимоисключающими.
Вы можете также указать прокси-сервер socks4 с помощью -x, --proxy, используя префикс протокола `socks4://`.
Можно использовать опцию --preproxy для указания прокси-сервера SOCKS одновременно задав опцией -x, --proxy использование прокси-сервера HTTP/HTTPS. В таком случае curl сначала подключается к прокси-серверу SOCKS, а затем подключается (через SOCKS) к прокси-серверу HTTP или HTTPS.
Если эта опция указана несколько раз, будет использоваться последняя.

--socks4a <хост[:порт]>

Задает указанный прокси-сервер SOCKS4a. Если номер порта не указан, предполагается, что он находится в порту 1080.
Этот параметр переопределяет любое предыдущее использование -x, --proxy, поскольку они являются взаимоисключающими.
Вы можете также указать прокси-сервер socks4 с помощью -x, --proxy, используя префикс протокола `socks4a://`.
Можно использовать опцию --preproxy для указания прокси-сервера SOCKS одновременно задав опцией -x, --proxy использование прокси-сервера HTTP/HTTPS. В таком случае curl сначала подключается к прокси-серверу SOCKS, а затем подключается (через SOCKS) к прокси-серверу HTTP или HTTPS.
Если эта опция указана несколько раз, будет использоваться последняя.

--socks5-basic

Указание curl использовать проверку подлинности имени пользователя/пароля при подключении к прокси-серверу SOCKS5. Аутентификация по имени пользователя/паролю включена по умолчанию. Используйте --socks5-gssapi для принудительной проверки подлинности GSS-API для прокси-серверов SOCKS5.

--socks5-gssapi-nec

В рамках согласования GSS-API согласовывается режим защиты. В стандарте RFC 1961, в разделе `4.3/4.4` написано, что он должен быть защищен, но реализация NEC этого не делает. Опция --socks5-gssapi-nec позволяет незащищенный обмен в процессе согласования режима защиты.

--socks5-gssapi-service <имя>

По умолчанию для SOCKS сервера используется служба socks-rcmd/server-fqdn. Эта опция позволяет вам задать другое имя службы.
Примеры:
`"--socks5 proxy-name --socks5-gssapi-service sockd"` задаст имя `"sockd/proxy-name"`,
`"--socks5 proxy-name --socks5-gssapi-service sockd/real-name"` задаст имя `"sockd/real-name"`
для случаев, когда имя прокси-сервера не совпадает с именем службы.

--socks5-gssapi

Указывает curl использовать аутентификацию GSS-API при подключении к прокси-серверу SOCKS5. Используйте --socks5-basic для принудительной проверки подлинности имени пользователя/пароля для прокси-серверов SOCKS5.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

--socks5-hostname <хост[:порт]>

Задает указанный прокси-сервер SOCKS5, при этом прокси-сервер автоматически определяет имя хоста. Если номер порта не указан, предполагается, что он находится в порту 1080.

Этот параметр переопределяет любое предыдущее использование -x, --proxy, поскольку они являются взаимоисключающими.

Вы можете также указать прокси-сервер socks4 с помощью -x, --proxy, используя префикс протокола `socks5h://`.

Можно использовать опцию --preproxy для указания прокси-сервера SOCKS одновременно задав опцией -x, –proxy использование прокси-сервера HTTP/HTTPS. В таком случае curl сначала подключается к прокси-серверу SOCKS, а затем подключается (через SOCKS) к прокси-серверу HTTP или HTTPS.

Если эта опция указана несколько раз, будет использоваться последняя.

--socks5 <хост[:порт]>

Задает указанный прокси-сервер SOCKS5 с указанием имени хоста. Если номер порта не указан, предполагается, что он находится в порту 1080.

Этот параметр переопределяет любое предыдущее использование -x, --proxy, поскольку они являются взаимоисключающими.

Вы можете также указать прокси-сервер socks4 с помощью -x, --proxy, используя префикс протокола `socks5://`.

Можно использовать опцию --preproxy для указания прокси-сервера SOCKS одновременно задав опцией `"-x, --proxy"` использование прокси-сервера HTTP/HTTPS. В таком случае curl сначала подключается к прокси-серверу SOCKS, а затем подключается (через SOCKS) к прокси-серверу HTTP или HTTPS.

Если эта опция указана несколько раз, будет использоваться последняя.

Этот параметр (также как --socks4) не работает с IPV6 и LDAP.

-Y, --speed-limit <скорость>

Если загрузка происходит медленнее, чем эта заданная скорость (в байтах в секунду) в течение нескольких секунд, она прерывается. Период времени задается с помощью -y, --speed-time и равна 30, если не указана явно.

Если эта опция указана несколько раз, будет использоваться последняя.

-y, --speed-time <секунды>

Если загрузка происходит медленнее чем указано в течение заданного периода, то загрузка прерывается. Если используется speed-time, ограничение скорости по умолчанию будет равно 1, если не установлено с помощью -Y, --speed-limit.

Этот параметр управляет передачей и, таким образом, не влияет на медленные соединения и т. д. В случае проблем с соединением попробуйте опцию --connect-timeout.

Если эта опция указана несколько раз, будет использоваться последняя.

--ssl-reqd

(FTP IMAP POP3 SMTP) Требовать соединения по SSL/TLS. Обрывает соединение, если сервер не поддерживает SSL/TLS.

--ssl

(FTP IMAP POP3 SMTP) По возможности использовать SSL/TLS для подключения. Если сервер не поддерживает SSL/TLS, то будет использоваться незащищенное соединение. См. также --ftp-ssl-control и --ssl-reqd для различных уровней шифрования.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

--stderr <файл>

Перенаправляет все ошибки в заданный файл вместо stderr. Если имя файла указано как тире "-", содержимое перенаправляется в stdout.
Если эта опция указана несколько раз, будет использоваться последняя.
См. также -v, --verbose и -s, --silent.

--styled-output

Включает автоматическое использование стилей полужирного шрифта при выводе заголовков HTTP в терминал. Используйте --no-style-output, чтобы отключить их.

--suppress-connect-headers

Когда используется -p, --proxytunnel и выполняется запрос CONNECT, заголовки ответов прокси CONNECT не будут выводиться. Этот параметр предназначен для использования с -D, --dump-header или -i, --include, которые используются для отображения заголовков протокола в выходных данных. Это не влияет на параметры отладки, такие как -v, --verbose или --trace, а также любую статистику.
См. также -D, --dump-header, -i, --include и -p, --proxytunnel.

--tcp-fastopen

Включает использование TCP Fast Open (Стандарт RFC7413).

--tcp-nodelay

Включает опцию TCP_NODELAY.
Эта опция включена по умолчанию, и вам нужно явно отключить ее, если вы не хотите, чтобы она использовалась.

-t, --telnet-option <имя=значение>

Передает параметры протоколу telnet. Поддерживаемые параметры:
TTYPE=<терминал> Задает тип терминала.
XDISPLOC=<X дисплей> Задает местоположение X - дисплея.
NEW_ENV=<имя,значение> Задает переменную окружения.

--tftp-blksize <значение>

(TFTP) Задает параметр TFTP BLKSIZE (должен быть >512). Это размер блока, который curl попытается использовать при передаче данных на сервер TFTP или с него. По умолчанию будет использоваться 512 байт.
Если эта опция указана несколько раз, будет использоваться последняя.

--tftp-no-options

(TFTP) Указывает, что curl не должен отправлять запрос параметров TFTP.
Этот параметр улучшает взаимодействие с некоторыми устаревшими серверами, которые должным образом не реализуют параметры TFTP. При использовании этой опции --tftp-blksize игнорируется.

-z, --time-cond <дата-время>

(HTTP FTP) Запрашивает файл, который был изменен позже заданного времени и даты, или файл, который был изменен до этого времени. Выражение <дата> может быть в любом формате даты, если оно не совпадает ни с какими форматами, то будет рассматриваться как имя файла и вместо этого пытается получить дату изменения (mtime) файла <файла>.
Укажите перед датой знак минус (-), чтобы запросить документ, который старше заданной даты/времени, по умолчанию запрашивается документ,
который новее указанной даты/времени.

Если эта опция указана несколько раз, будет использоваться последняя.

--tls-max <ВЕРСИЯ>

(TLS) <ВЕРСИЯ> определяет максимально допустимую версию TLS. Минимально допустимая версия устанавливается параметрами --tlsv1.0, --tlsv1.1, --tlsv1.2 или --tlsv1.3. Если соединение выполняется без TLS, этот параметр не имеет никакого эффекта. Это включает в себя передачу с использованием QUIC (HTTP/3).

default используется наивысшая рекомендуемая версия TLS.

1.0 используется максимум TLS 1.0.

1.1 используется максимум TLS 1.1.

1.2 используется максимум TLS 1.2.

1.3 используется максимум TLS 1.3.

См. также --tlsv1.0, --tlsv1.1, --tlsv1.2 и --tlsv1.3.

--tls13-ciphers <список криптонаборов>

(TLS) Указывает, какие криптонаборы использовать в соединении TLS версий 1.3. Для TLS версии 1.2 следует использовать опцию --ciphers.
Поддерживаются следующие российские криптонаборы:
`TLS_GOSTR341112_256_WITH_MAGMA_MGM_S`,
`TLS_GOSTR341112_256_WITH_MAGMA_MGM_L`,
`TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_S` и
и `TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_L`.
Если эта опция используется несколько раз, будет использоваться последняя.

--tlsauthtype <тип>

Указывает тип проверки подлинности TLS. В настоящее время единственным поддерживаемым вариантом является `"SRP"` для TLS-SRP (RFC 5054). Если указаны --tlsuser и --tlspassword, а --tlsauthtype не указан, то по умолчанию этот параметр имеет значение `"SRP"`.

--tlspassword

Задает пароль для использования с методом проверки подлинности TLS, указанным с помощью --tlsauthtype. Требует, чтобы --tlsuser также был установлен.
Неприменимо с TLS 1.3.

--tlsuser <имя>

Задает имя пользователя для использования с методом проверки подлинности TLS, указанным с помощью --tlsauthtype. Требуется, чтобы --tlspassword также был установлен.
Неприменимо с TLS 1.3.

--tlsv1.0

(TLS) Указывает, что curl должен использовать TLS версии 1.0 или более поздней при подключении к удаленному серверу TLS.
Используйте --tls-max, если вы хотите установить максимальную версию TLS.

--tlsv1.1

(TLS) Указывает, что curl должен использовать TLS версии 1.1 или более поздней при подключении к удаленному серверу TLS.
Используйте --tls-max, если вы хотите установить максимальную версию TLS.

--tlsv1.2

(TLS) Указывает, что curl должен использовать TLS версии 1.2 или более поздней при подключении к удаленному серверу TLS.

Используйте --tls-max, если вы хотите установить максимальную версию TLS.

--tlsv1.3

(TLS) Указывает, что curl должен использовать TLS версии 1.3 или более поздней при подключении к удаленному серверу TLS.

Если соединение выполняется без TLS, этот параметр не имеет никакого эффекта. Это включает в себя передачу с использованием QUIC (HTTP/3).

-1, --tlsv1

(TLS) Указывает curl использовать по крайней мере TLS версии 1.x при согласовании с удаленным сервером TLS (TLS 1.0 или выше).

См. также --http1.1 и --http2. Этот параметр переопределяет --tlsv1.1 и --tlsv1.2 и --tlsv1.3.

--tr-encoding

(HTTP) Запрашивает сжатый ответ Transfer-Encoding, используя один из алгоритмов, поддерживаемых curl, и распаковывает данные во время их получения.

--trace-ascii <файл>

Включает полный дамп трассировки всех входящих и исходящих данных, включая описательную информацию, в данный выходной файл. Используйте `"-"` в качестве имени файла для отправки выходных данных в stdout.

Поведение похоже на --trace, но отбрасывает шестнадцатеричную часть и показывает только ASCII-часть дампа. В результате вывод получается меньше и может быть проще для чтения.

Если эта опция указана несколько раз, будет использоваться последняя.

Этот параметр переопределяет --trace и -v, --verbose.

--trace-time

Добавляет метку времени к каждой трассировке или подробной строке, отображаемой curl.

--trace <файл>

Сохраняет полный дамп трассировки всех входящих и исходящих данных, включая описательную информацию, в данный выходной файл. Используйте знак минус (-) в качестве имени файла для отправки выходных данных в stdout. Используйте "

Если эта опция указана несколько раз, будет использоваться последняя.

Этот параметр переопределяет -v, --verbose и --trace-ascii.

--unix-socket <путь>

(HTTP) Указание подключаться через указанный сокет домена Unix вместо использования сети.

-T, --upload-file <файл>

Загружает указанный локальный файл на удаленный URL-адрес. Если в указанном URL-адресе не указано имя файла curl добавит имя локального файла. ВНИМАНИЕ: вы должны указать завершающий слеш "/"в последнем каталоге, чтобы curl понял, что имя файла не указано. В противном случае curl расценит последнее имя каталога как имя удаленного файла, что скорее всего приведет к ошибке загрузки. Если опция используется на сервере HTTP(S), то будет использоваться команда PUT.

Используйте имя файла `"-"` (тире) чтобы считывать данные из stdin вместо файла.

Используйте имя файла "." (точка), чтобы использовать stdin в неблокирующем режиме и разрешить чтение выходных данных сервера во время загрузки stdin.
Вы можете указать опцию -T, --upload-file для каждого URL-адреса в командной строке. Каждая пара -T, --upload-file + URL указывает, что и куда загружать. curl также поддерживает загрузку нескольких файлов по одному URL-адресу, например:

```
curl --upload-file "{file1,file2}" http://www.example.com
```

или

```
curl -T "img[1-1000].png" ftp://ftp.example.com/upload/
```

При загрузке на SMTP-сервер предполагается, что загруженные данные отформатированы в формате RFC 5322. Данные должны содержать необходимый набор заголовков и тело сообщения в корректном формате, так как curl не будет ни перекодировать, ни кодировать данные каким-либо образом.

--url <url>

Указать URL для скачивания. Этот параметр в основном полезен, когда вы хотите указать URL в файле конфигурации.
Если в данном URL отсутствует имя схемы (например, "http://"или "ftp://"и т. д.), curl сделает предположение на основе имени хоста. Если внешнее поддоменное имя соответствует DICT, FTP, IMAP, LDAP, POP3 или SMTP, то будет использоваться этот протокол, иначе будет использоваться HTTP. Автоопределение можно отключить, установив протокол по умолчанию, подробности см. в разделе --proto-default.
Эта опция может использоваться любое количество раз. Чтобы указать, куда запишется скачанный файл, используйте параметры -o, --output или -O, --remote-name.
Предупреждение: В Windows доступ к file:// может быть преобразован операционной системой в доступ к сетевому ресурсу. Будьте внимательны!

-B, --use-ascii

(FTP, LDAP) Использовать передачу в ASCII. Для FTP передачу в ASCII можно включить, указав URL, который заканчивается на ";type=A". В win32-системах это приводит к тому, что данные, отправляемые в stdout, находятся в текстовом режиме.

-A, --user-agent <имя>

(HTTP) Указывает строку User-Agent, которая будет передана на сервер HTTP. Если строка содержит пробелы, заключите ее в кавычки. Этот заголовок также можно задать с помощью параметров -H, --header или --proxy-header.
Если вы укажете пустой аргумент -A, --user-agent `("")`, это полностью удалит заголовок User-Agent из запроса. Если вам нужно, чтобы этот заголовок был, но пустой, установите его в пробел `(" ")`.
Если эта опция указана несколько раз, будет использоваться последняя.

-u, --user <пользователь:пароль>

Указать имя пользователя и пароль для аутентификации на сервере. Переопределяет значения, указанные в -n, --netrc и --netrc-optional
Если вы укажете только имя пользователя, curl запросит пароль.
Имя пользователя отделяется от пароля первым встретившимся двоеточием, что делает невозможным с этой опцией использовать двоеточия в имени пользователя. Пароль может содержать двоеточие.
В тех операционных системах, где это работает, curl старается скрыть этот аргумент из списка процессов. Но они все равно будут видны какое-то время, прежде чем curl сумеет их спрятать, так что это небезопасно. Такие конфиденциальные данные не следует

использовать в открытом тексте в командной строке, лучше читать их из файла.
При использовании Kerberos V5 в Windows следует включить имя домена Windows в имя пользователя, чтобы сервер успешно получил Kerberos Ticket. Если вы этого не сделаете, то первоначальное аутентификационный хэндшейк может завершиться неудачно.
При использовании NTLM имя пользователя можно указать просто без домена, если, например, в вашей конфигурации только один домен и лес.
Чтобы указать доменное имя, используйте форматы Down-Level Logon Name или UPN. Например, `EXAMPLE\\user` и user@example.com соответственно.
Если эта опция указана несколько раз, будет использоваться последняя.

-v, --verbose

Включает подробные сообщения curl. Это полезно для отладки и просмотра того, что происходит "под капотом". Строка, начинающаяся с `">"`, означает служебную информацию, отправленную curl, `"<"` означает служебную информацию, полученную curl, которая в обычных случаях не видна пользователю, а строка, начинающаяся с `"*"`, означает дополнительную информацию, предоставленную curl.
Если вам нужны только заголовки HTTP в выходных данных, вам подойдет -i, --include.
Если вы считаете, что с этим параметром вам все еще не хватает подробностей, попробуйте использовать --trace или --trace-ascii.
Используйте -s, --silent, чтобы сделать curl по-настоящему молчаливым.
См. также -i, --include. Этот параметр переопределяет --trace и --trace-ascii.

-V, --version

Отображает информацию о curl и используемой версии libcurl.
Первая строка включает полную версию curl, libcurl и версии других библиотек, с которыми собран исполняемый файл.
Вторая строка (начинается с "Protocols:") отображает все протоколы, которые поддерживает данная libcurl.
Третья строка (начинается с "Features:") показывает конкретные возможности, которые поддерживает умеет данная libcurl. Доступные функции:

alt-svc

Предусмотрена поддержка Alt-Svc.

AsynchDNS

Использует асинхронные разрешения имен. Асинхронное разрешение имен может быть реализовано либо с помощью библиотеки c-ares, либо с использованием тредов.

HTTPS-proxy

curl собран с поддержкой HTTPS-прокси.

IDN

curl поддерживает IDN - доменные имена с символами национальных алфавитов

IPv6

curl собран с поддержкой IPv6.

Kerberos

Поддерживается аутентификация Kerberos V5.

Largefile

curl поддерживает передачу больших файлов, файлов размером более 2 ГБ.

libz
Поддерживается автоматическая распаковка (gzip, deflate) сжатых файлов по протоколу HTTP.

MultiSSL
Этот curl поддерживает несколько бэкендов TLS.

NTLM
Поддерживается аутентификация NTLM.

NTLM_WB
Поддерживается делегирование NTLM Winbind Helper

SPNEGO
Поддерживается аутентификация SPNEGO.

SSL
Поддерживаются SSL для различных протоколов, таких как HTTPS, POP3S и так далее.

SSPI
Поддерживается SSPI.

TLS-SRP
Аутентификация SRP (Secure Remote Password) поддерживается для TLS.

Unicode
Поддержка Unicode в Windows.

UnixSockets
Предусмотрена поддержка Unix Sockets.

-w, --write-out <format>
Заставляет curl отображать информацию в stdout после завершения загрузки. Формат - строка, которая может содержать обычный текст, и любое количество переменных. Формат может быть указан как литеральная "строка или вы можете попросить curl прочитать формат из файла с "@filename"и сказать curl прочитать формат из stdin, для чего нужно указать "@-".
Переменные, присутствующие в выходном формате, будут заменены значениями, подставленными curl, как описано ниже. Все переменные задаются как `%{variable_name}`. Если вам нужен `%`, в строке формата укажите вы `%%`. Новая строка обозначается как `\n`, возврат каретки `\r`, табуляции `\t`.
Вывод будет записан на stdout, но его можно переключить на stderr с помощью `%{stderr}`.
ПРИМЕЧАНИЕ: Символ `%`-это специальный символ в среде win32, где все вхождения `%` должны быть удвоены при использовании этой опции.
Доступны следующие переменные:

content_type
Content-Type запрошенного документа, если таковой имеется.

filename_effective
Имя файла, в которое curl сохранил загрузку. Имеет смысл только если curl будет указано имя файла для записи, параметром -O, --remote-name или -o, --output. Наиболее полезно в сочетании с опцией -J, --remote-header-name.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

ftp_entry_path
Начальный путь, по которому curl оказался при входе на удаленный FTP-сервер.

http_code
Числовой код завершения, который был получен при последней загрузке HTTP(S) или FTP(s). Синонимом этой переменной является response_code.

http_connect
Числовой код, который был получен в последнем ответе от прокси-сервера на запрос CONNECT.

http_version
Реально использовавшаяся при загрузке версия http

json
Объект JSON со всеми доступными ключами.

local_ip
IP - локальный ip-адрес при последнем соединении. Может быть либо IPv4, либо IPv6.

local_port
Локальный порт при последнем соединении.

method
Метод http, используемый в последнем HTTP-запросе.

num_connects
Количество новых подключений, сделанных в ходе недавней передачи.

num_headers
Количество заголовков ответа в последнем запросе (перезапускается при каждом редиректе). Обратите внимание, что строка состояния НЕ ЯВЛЯЕТСЯ заголовком.

num_redirects
Количество редиректов, сделанных при последнем запросе.

proxy_ssl_verify_result
Результат проверки TLS-сертификата HTTPS-прокси. 0 означает, что проверка прошла успешно.

redirect_url
Когда HTTP-запрос был сделан без -L, --location для отслеживания перенаправлений (или когда выполняется --max-redir), эта переменная покажет фактический URL-адрес, на который должно было перейти перенаправление.

remote_ip
Удаленный IP-адрес при последнем подключении. Может быть либо IPv4, либо IPv6.

remote_port
Номер удаленного порта при последнем подключении.

response_code
Числовой код завершения при последней загрузке. Синоним "http_code".

scheme
Схема URL (иногда называемая протоколом), которая использовалась при последней загрузке.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

size_download
Общее количество загруженных байтов.

size_header
Общее количество байтов загруженных заголовков.

size_request
Общее количество байтов, отправленных в HTTP-запросе.

size_upload
Общее количество загруженных байтов.

speed_download
Средняя скорость скачивания, измеренная curl, байт в секунду.

speed_upload
Средняя скорость загрузки на сервер, измеренная curl, байт в секунду.

ssl_verify_result
Результат проверки TLS-сертификата сервера, 0 в случае успеха.

stderr
С этого момента вывод -w, --write-out будет записан в stderr.

stdout
С этого момента вывод -w, --write-out будет записан на stdout. Это значение по умолчанию, но его можно использовать для переключения обратно после переключения на stderr.

time_appconnect
Время в секундах, с момента запуска curl до завершения TLS-хендшейка.

time_connect
Время в секундах, с момента запуска curl до установленного TCP-соединения с удаленным хостом (или прокси-сервером).

time_namelookup
Время в секундах, которое потребовалось с момента запуска curl до завершения разрешения имени.

time_pretransfer
Время в секундах, которое потребовалось с момента запуска curl до начала передачи файлов. Включает в себя все команды предварительной передачи и переговоры, специфичные для конкретных протоколов.

time_redirect
Время в секундах, которое потребовалось для всех шагов перенаправления, включая разрешение имен, подключение, предварительную передачу, прежде чем была запущена окончательная транзакция. time_redirect показывает полное время выполнения для нескольких перенаправлений.

time_starttransfer
Время в секундах, которое потребовалось с самого начала, пока первый байт не был передан. Это включает в себя time_pretransfer, а также время, необходимое серверу для вычисления результата.

time_total
Общее время, в секундах, в течение которого длилась полная операция.

url_effective
URL-адрес, который был выбран последним. Это наиболее важно, если вы сказали curl следовать за location:headers.

Если опция --write-out написана несколько раз, будет использоваться последняя.

--xattr
При сохранении скачанных данных в файл, этот параметр указывает curl сохранять метаданные файла в расширенных атрибутах файла (xattr). Сейчас URL сохраняется в атрибуте xdg.origin.url, a content type для HTTP сохраняется в атрибуте mime_type. Если файловая система не поддерживает xattr, выдается предупреждение.

6.7 Переменные окружения

Переменные среды могут быть указаны в нижнем или верхнем регистре. Указанные в нижнем регистре имеют приоритет. http_proxy является исключением, поскольку он доступен только в нижнем регистре.

Использование переменной среды для установки прокси-сервера имеет тот же эффект, что и использование параметра -x, --proxy.

`http_proxy` [protocol://]<host>[:port]
Задает прокси - сервер, используемый для HTTP.

`HTTPS_PROXY` [protocol://]<host>[:port]
Устанавливает прокси-сервер для использования по протоколу HTTPS.

`[url-protocol]_PROXY` [protocol://]<host>[:port]
Задает прокси-сервер для использования для [url-protocol], где protocol - это протокол, поддерживаемый curl и указанный в URL. FTP, POP3, IMAP, SMTP, LDAP и т.д.

`ALL_PROXY` [protocol://]<host>[:port]
Задает прокси-сервер для использования, если не установлен прокси-сервер для конкретного протокола.

`NO_PROXY` <список хостов/доменов, разделенных запятыми>
список имен хостов, для которых не должен использоваться прокси-сервер. Символ "*"соответствует всем хостам. Имена в этом списке могут быть либо именами доменов, либо именами конкретных хостов.

Эта переменная среды отключает использование прокси-сервера, даже если он указан с параметром -x, --proxy. То есть

```
NO_PROXY=direct.example.com curl -x http://proxy.example.com \
  http://direct.example.com
```

получает прямой доступ к целевому URL и

```
NO_PROXY=direct.example.com curl -x http://proxy.example.com \
  http://somewhere.example.com
```

получает доступ к целевому URL через прокси-сервер.

Список имен хостов также может включать числовые IP-адреса. Адреса IPv6 должны указываться без скобок.

Числовые адреса IPv6 сравниваются как строковые значения, поэтому они будут совпадать только в том случае, если представления одинаковы: `"::1"` и `"::0:1"` будут считаться разными адресами.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

`SSLKEYLOGFILE` <имя файла>

Если вы зададите этой переменной окружения имя файла, curl будет сохранять TLS secrets своих подключений в этом файле, чтобы вы могли анализировать трафик TLS в режиме реального времени с помощью инструментов анализа сети, таких как Wireshark.

6.8 Префиксы протокола прокси

Строка прокси-сервера может быть указана с префиксом protocol:// для указания альтернативных протоколов прокси-сервера.

Если в строке прокси-сервера не указан протокол или если строка не соответствует поддерживаемому протоколу, прокси-сервер будет рассматриваться как HTTP-прокси.

Поддерживаемые префиксы протокола прокси-сервера следующие:

http://

Используется HTTP-прокси. Значение по умолчанию, если префикс схемы не используется.

https://

Используется HTTPS-прокси.

socks4://

Эквивалентно --socks4

socks4a://

Эквивалентно --socks4a

socks5://

Эквивалентно --socks5

socks5h://

Эквивалентно --socks5-hostname

6.9 Коды ошибок

Если что-то пошло не так, curl возвращает ненулевой код завершения. Кодов ошибок много, и их список пополняется от версии к версии. На данный момент определены следующие:

1

Текущая сборка curl не поддерживает этот протокол.

2

Не удалось инициализировать curl

3

Неправильный синтаксис URL

4

Используется функция или параметр, не поддерживаемый в данной сборке.

5

Не удалось определить адрес прокси-сервера

6

Не удалось определить адрес хоста.

7

Не удалось подключиться к хосту.

8

Сервер отправил данные, которые curl не смог распарсить.

9

Доступ по FTP запрещен. Сервер отказал во входе в систему или в доступе к конкретному ресурсу или каталогу, к которому вы хотели обратиться. Обычно это означает, что вы пытались перейти в каталог, который не существует на сервере.

10

Ошибка FTP в активном режиме. Во время ожидания обратного подключения сервера FTP, по управляющему соединению был отправлен код ошибки.

11

Ошибка FTP. Curl не понял, что отправил сервер в ответ на отправку пароля (командой PASS)

12

Ошибка FTP в активном режиме. При ожидании обратного соединения от сервера достигнут тайм-аут.

13

Ошибка FTP при запросе пассивного режима. Curl отправил команду PASV, но не понял, что ответил сервер.

14

Ошибка FTP при запросе пассивного режима. Curl отправил команду PASV, сервер ответил строкой 227 (переход в пассивный режим), но curl не сумел ее распарсить.

15

Ошибка FTP в пассивном режиме. Сервер отправил строку 227, но curl не сумел разрезолвить адрес переданного хоста.

16

Ошибка HTTP2 framing layer.

17

FTP не смог перейти в двоичный режим передачи данных.

18

Файл был передан не полностью.

19

FTP не смог загрузить/получить доступ к данному файлу, команда RETR (или аналогичная) не удалась.

21

Ошибка FTP. Сервер вернул ошибку после команды QUOTE.

22

HTTP-страница не получена. Запрошенный URL не был найден, или сервер вернул другую ошибку HTTP с кодом 400 или выше. Этот код возврата появляется только в том случае, если используется -f, --fail.

23

Ошибка записи. Curl не мог записать данные в локальную файловую систему.

25

Ошибка FTP. Сервер запретил операцию загрузки на него файла (команду STOR)

26

Ошибка чтения. Различные проблемы с чтением.

27

Недостаточно памяти. Не удалось выполнить запрос на выделение памяти.

28

Тайм-аут операции.

30

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Ошибка FTP. Сервер не смог выполнить команду PORT (вход в активный режим). Попробуйте использовать пассивный режим (команда PASV)

31

Ошибка FTP, не удалось выполнить команду REST, которая используется при докачке файла.

33

Ошибка HTTP. `"Команда"` range не сработала.

34

Ошибка HTTP. Внутренняя ошибка генерации POST-запроса.

35

Ошибка подключения TLS. TLS-хэндшейк не удался.

36

Не удалось продолжить ранее прерванную загрузку.

37

Не удалось прочитать файл.

38

Не удалось подключиться к LDAP.

39

Поиск LDAP не удался.

41

Требуемая функция LDAP не была найдена.

42

Неуспешный callback. Операция curl прервана по команде приложения.

43

Внутренняя ошибка. Функция была вызвана с неправильным параметром.

45

Ошибка интерфейса. Указанный исходящий интерфейс не может быть использован.

47

Слишком много перенаправлений.

48

Неизвестный параметр, указанный в libcurl. curl получил опцию, которая была передана libcurl и отклонена.

49

Неправильный формат вызова telnet.

51

TLS-сертификат или SSH fingerprint удаленного хоста неправильный.

52

Сервер ничего не ответил, в тех обстоятельствах, когда это ошибка.

53

Ошибка TLS: подгружаемый модуль (engine) не найден.

54

Ошибка TLS: не удается установить подгружаемый модуль (engine) по умолчанию.

55

Не удалось отправить данные по сети.

56

Не удалось получить данные по сети.

58

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Проблема с локальным сертификатом.
59
Не удалось использовать указанный криптонабор TLS.
60
Сертификат удаленного хоста не может быть проверен с помощью известных сертификатов CA.
61
Нераспознанная кодировка передачи.
62
Недопустимый URL LDAP.
63
Превышен максимальный размер файла.
64
Ошибка FTP. Не удалось соединиться с сервером с запрошенными настройками TLS
65
Для отправки данных требуется `"перемотка назад"`, которая не удалась.
66
Не удалось инициализировать OpenSSL.
67
Curl не удалось аутентифицироваться.
68
Файл не найден на сервере TFTP.
69
Проблема с правами доступа на сервере TFTP.
70
На сервере TFTP не хватает места на диске.
71
Неправильная операция TFTP.
72
Неизвестный TFTP transfer ID.
73
Файл уже существует (TFTP).
74
Нет такого пользователя (TFTP).
75
Преобразование символов не удалось.
76
Требуются функции преобразования символов.
77
Проблема с чтением сертификата TLS CA (путь? права доступа?).
78
Ресурс, на который ссылается URL, не существует.
79
Во время сеанса SSH произошла неопределенная ошибка.
80
Не удалось завершить TLS-соединение.
82

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Не удалось загрузить файл CRL, он отсутствует, или у него неправильный формат.
83
Не удалось проверить издателя сертификата.
84
Ошибка FTP, не удалось выполнить команду PRET
85
RTSP: несоответствие номеров CSeq
86
RTSP: несоответствие идентификаторов сеансов
87
Не удалось распарсить список файлов FTP
88
Ошибка FTP. Сообщение о неудачном вызове chunk callback
89
Соединение недоступно, сеанс будет поставлен в очередь
90
Открытый ключ TLS не соответствует закрепленному открытому ключу
91
Недопустимый статус TLS-сертификата.
92
Ошибка потока в слое кадрирования HTTP/2.
93
Callback к функции API
94
Функция аутентификации вернула ошибку.
95
Ошибка HTTP/3.
96
Ошибка подключения QUIC.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий сопроводительного докум. и дата	Подпись	Дата
Изм.	измененных	замененных	новых	аннулированных					

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения