

curl クイックリファレンス

HTTP リクエスト、ヘッダー、認証、フォーム、デバッグ

基本的な使い方

シンプルなリクエスト

```
curl https://example.com # GET リクエスト
curl -o file.html https://url # ファイルに保存
curl -O https://url/file.tar.gz # リモート名で保存
curl -L https://url # リダイレクトに従う
```

よく使うフラグ

| | |
|----------------------|--------------------|
| -s | サイレントモード (進捗なし) |
| -S | サイレントモードでもエラーを表示 |
| -f | HTTP エラー時にサイレントで失敗 |
| -L | リダイレクトに従う |
| -o file | 出力をファイルに書き込む |
| -O | リモートのファイル名で保存 |
| -C - | 中断されたダウンロードを再開 |
| --max-time 30 | 30 秒後にタイムアウト |

HTTP メソッド

GET と HEAD

```
curl https://api.example.com/users
curl -I https://example.com # HEAD (ヘッダーのみ)
curl -i https://example.com # レスポンスヘッダーを含む
```

POST

```
curl -X POST https://api.example.com/users \
-H "Content-Type: application/json" \
-d '{"name":"Jo","email":"jo@ex.com"}'
```

PUT と PATCH と DELETE

```
curl -X PUT https://api.example.com/users/1 \
-d '{"name":"Updated"}'
curl -X PATCH https://api.example.com/users/1 \
-d '{"email":"new@ex.com"}'
curl -X DELETE https://api.example.com/users/1
```

ヘッダー

ヘッダーの設定

```
curl -H "Content-Type: application/json" URL
curl -H "Accept: text/html" URL
curl -H "X-Custom: value" URL
curl -H "Header1: v1" -H "Header2: v2" URL
```

レスポンスヘッダー

| | |
|--------------------------|-------------------|
| -i | 出力にレスポンスヘッダーを含める |
| -I | ヘッダーのみ取得 (HEAD) |
| -D file | レスポンスヘッダーをファイルに保存 |
| -w '%{http_code}' | HTTP ステータスコードを表示 |

認証

Basic 認証とトークン認証

```
curl -u user:pass https://api.example.com
curl -H "Authorization: Bearer TOKEN" URL
curl -u user:pass --digest URL
curl --negotiate -u : URL # Kerberos/SPNEGO
```

認証方式

| | |
|---------------------|--------------------|
| -u user:pass | Basic 認証 |
| --digest | HTTP Digest 認証 |
| --negotiate | Kerberos/SPNEGO 認証 |
| --ntlm | NTLM 認証 |
| -n | ~/netrc の認証情報を使用 |

データとフォーム

データの送信

```
curl -d "key=val&key2=val2" URL # フォーム URL エンコード
curl -d @data.json URL # ファイルからデータを送信
curl --data-raw '{"raw":"json"}' URL
curl --data-urlencode "q=hello world" URL
```

ファイルのアップロード

```
curl -F "file=@photo.jpg" URL
curl -F "file=@doc.pdf;type=application/pdf" URL
curl -F "field=value" -F "file=@img.png" URL
```

マルチパート vs URL エンコード

| | |
|----------------|---------------------------------------|
| -d | application/x-www-form-urlencoded |
| -F | multipart/form-data |
| --json | 短縮形: Content-Type と Accept を JSON に設定 |
| -T file | PUT でファイルをアップロード |

SSL/TLS

証明書オプション

```
curl --cacert ca.pem URL # カスタム CA バンドル
curl --cert client.pem URL # クライアント証明書
curl --cert client.pem --key key.pem URL
curl -k URL # TLS 検証をスキップ (開発時のみ)
```

TLS フラグ

| | |
|------------------------|-----------------|
| -k / --insecure | TLS 証明書の検証をスキップ |
| --cacert file | カスタム CA 証明書を使用 |
| --cert file | クライアント証明書 |
| --key file | クライアント秘密鍵 |
| --tlsv1.2 | 最小 TLS 1.2 を強制 |
| --tlsv1.3 | 最小 TLS 1.3 を強制 |

出力とデバッグ

詳細出力とトレース

```
curl -v URL # 詳細出力
curl --trace dump.txt URL # ファイルにフルトレース
curl --trace-ascii - URL # 標準出力にトレース
curl -w "\n%{http_code}\n" URL # カスタム出力フォーマット
```

Write-Out 変数

| | |
|-----------------------------|--------------------|
| %{http_code} | HTTP レスポンスステータスコード |
| %{time_total} | 合計時間 (秒) |
| %{time_connect} | 接続確立までの時間 |
| %{size_download} | ダウンロードしたバイト数 |
| %{speed_download} | 平均ダウンロード速度 |
| %{redirect_url} | リダイレクト URL (あれば) |
| %{ssl_verify_result} | SSL 検証結果 (0=OK) |

Write-Out の例

```
curl -s -o /dev/null -w \
"code: %{http_code}\ntime: %{time_total}s\n" \
https://example.com
```

よくあるパターン

API ワークフロー

```
# JSON を取得して jq にパイプ
curl -s https://api.example.com/data | jq '.items[]'
# 認証付きで JSON を POST
curl -s -H "Authorization: Bearer $TOKEN" \
--json '{"key":"val"}' https://api.example.com
```

ダウンロードパターン

```
# プログレスバー付きでダウンロード
curl -# -O https://releases.example.com/v2.tar.gz
# 中断されたダウンロードを再開
curl -C - -O https://releases.example.com/v2.tar.gz
# 複数ファイルをダウンロード
curl -O https://url/file1 -O https://url/file2
```

スクリプトのヘルパー

```
# URL に到達できるか確認
curl -sf -o /dev/null https://example.com && echo OK
# Cookie を保存して再利用
curl -c cookies.txt -b cookies.txt URL
# リクエストをレート制限
curl --limit-rate 100k URL
```

プロキシとネットワーク

プロキシ設定

```
curl -x http://proxy:8080 URL
curl -x socks5://proxy:1080 URL
curl --proxy-user user:pass -x http://proxy:8080 URL
curl --noproxy "*.local,localhost" URL
```

DNS と解決

| | |
|---------------------------------|----------------------|
| --resolve host:port:addr | DNS 解決を addr に強制 |
| --dns-servers 8.8.8.8 | カスタム DNS サーバーを使用 |
| --interface eth0 | 特定のネットワークインターフェースを使用 |
| -4 / -6 | IPv4 / IPv6 を強制 |

設定と応用

設定ファイル

```
# ~/.curlrc - デフォルトオプション
--silent
--location
--max-time 30
```

```
# 設定ファイルを明示的に使用
curl -K myconfig.txt URL
```

便利なフラグ

| | |
|--------------------------|-------------------|
| --retry 3 | 一時的なエラー時に再試行 |
| --retry-delay 2 | 再試行間の遅延 (秒) |
| --compressed | gzip/br を要求して展開 |
| --limit-rate 100k | 転送速度を制限 |
| -Z | 並列転送 (curl 7.66+) |
| --create-dirs | -o のパスディレクトリを作成 |