CURL 快速参考

HTTP 请求、头部、认证、表单、调试

基本用法

简单请求

```
curl https://example.com           # GET request
curl -o file.html https://url      # save to file
curl -O https://url/file.tar.gz    # save with remote name
curl -L https://url                # follow redirects
```

常用参数

`-s`	静默模式（不显示进度）
`-S`	静默模式下显示错误
`-f`	HTTP 错误时静默失败
`-L`	跟随重定向
`-o file`	输出写入文件
`-O`	使用远端文件名保存
`-C -`	续传中断的下载
`--max-time 30`	30 秒后超时

HTTP 方法

GET 与 HEAD

```
curl https://api.example.com/users
curl -I https://example.com       # HEAD (headers only)
curl -i https://example.com       # include response headers
```

POST

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

PUT、PATCH 与 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
```

请求头

设置请求头

```
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
```

响应头

`-i`	在输出中包含响应头
`-I`	仅获取响应头（HEAD）
`-D file`	将响应头写入文件
`-w '%{http_code}'`	打印 HTTP 状态码

认证

Basic 认证与 Token 认证

```
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
```

认证方式

`-u user:pass`	Basic 认证
`--digest`	HTTP Digest 认证
`--negotiate`	Kerberos/SPNEGO 认证
`--ntlm`	NTLM 认证
`-n`	使用 ~/.netrc 凭证

数据与表单

发送数据

```
curl -d "key=val&key2=val2" URL   # form urlencoded
curl -d @data.json URL            # data from file
curl --data-raw '{"raw":"json"}' URL
curl --data-urlencode "q=hello world" URL
```

文件上传

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

Multipart 与 URL 编码对比

`-d`	application/x-www-form-urlencoded
`-F`	multipart/form-data
`--json`	简写：设置 Content-Type + Accept 为 JSON
`-T file`	通过 PUT 上传文件

SSL/TLS

证书选项

```
curl --cacert ca.pem URL          # custom CA bundle
curl --cert client.pem URL        # client certificate
curl --cert client.pem --key key.pem URL
curl -k URL                       # skip TLS verify (dev only)
```

TLS 参数

`-k / --insecure`	跳过 TLS 证书验证
`--cacert file`	使用自定义 CA 证书
`--cert file`	客户端证书
`--key file`	客户端私钥
`--tlsv1.2`	强制最低 TLS 1.2
`--tlsv1.3`	强制最低 TLS 1.3

输出与调试

详细输出与追踪

```
curl -v URL                       # verbose output
curl --trace dump.txt URL         # full trace to file
curl --trace-ascii - URL          # trace to stdout
curl -w "\n%{http_code}\n" URL    # custom output format
```

Write-Out 变量

`%{http_code}`	HTTP 响应状态码
`%{time_total}`	总耗时（秒）
`%{time_connect}`	建立连接耗时
`%{size_download}`	下载字节数
`%{speed_download}`	平均下载速度
`%{redirect_url}`	重定向 URL（如有）
`%{ssl_verify_result}`	SSL 验证结果（0 = 正常）

Write-Out 示例

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

常用模式

API 工作流

```
# GET JSON and pipe to jq
curl -s https://api.example.com/data | jq '.items[]'
# POST JSON with auth
curl -s -H "Authorization: Bearer $TOKEN" \
  --json '{"key":"val"}' https://api.example.com
```

下载模式

```
# Download with progress bar
curl -# -O https://releases.example.com/v2.tar.gz
# Resume interrupted download
curl -C - -O https://releases.example.com/v2.tar.gz
# Download multiple files
curl -O https://url/file1 -O https://url/file2
```

脚本辅助

```
# Check if URL is reachable
curl -sf -o /dev/null https://example.com && echo OK
# Save cookies and reuse
curl -c cookies.txt -b cookies.txt URL
# Rate-limit request
curl --limit-rate 100k URL
```

代理与网络

代理设置

```
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 与解析

`--resolve host:port:addr`	强制 DNS 解析到指定地址
`--dns-servers 8.8.8.8`	使用自定义 DNS 服务器
`--interface eth0`	使用指定网络接口
`-4 / -6`	强制 IPv4 / IPv6

配置与进阶

配置文件

```
# ~/.curlrc — default options
--silent
--location
--max-time 30

# Use config file explicitly
curl -K myconfig.txt URL
```

实用参数

`--retry 3`	遇到临时错误时重试
`--retry-delay 2`	重试间隔（秒）
`--compressed`	请求并解压 gzip/br
`--limit-rate 100k`	限制传输速度
`-Z`	并行传输（curl 7.66+）
`--create-dirs`	为 -o 创建路径目录