## NAME

libcurl-url - URL interface overview

## DESCRIPTION

The URL interface provides functions for parsing and generating URLs.

## INCLUDE

You still only include <curl/curl.h> in your code.

## CREATE

Create a handle that holds URL info and resources with *curl_url(3)*:

```
CURLU *h = curl_url();
```

## CLEANUP

When done with it, clean it up with *curl_url_cleanup(3)*

```
curl_url_cleanup(h);
```

## DUPLICATE

When you need a copy of a handle, just duplicate it with *curl_url_dup(3)*:

```
CURLU *nh = curl_url_dup(h);
```

## PARSING

By setting a URL to the handle with *curl_url_set(3)*, the URL is parsed and stored in the handle. If the URL is not syntactically correct it returns an error instead.

```
rc = curl_url_set(h, CURLUPART_URL,
                  "https://example.com:449/foo/bar?name=moo", 0);
```

The zero in the fourth argument is a bitmask for changing specific features.

If successful, this stores the URL in its individual parts within the handle.

## REDIRECT

When a handle already contains info about a URL, setting a relative URL makes it "redirect" to that.

```
rc = curl_url_set(h, CURLUPART_URL, "../test?another", 0);
```

## GET URL

The **CURLU** handle represents a URL and you can easily extract that with *curl_url_get(3)*:

```
char *url;
rc = curl_url_get(h, CURLUPART_URL, &url, 0);
curl_free(url);
```

The zero in the fourth argument is a bitmask for changing specific features.

## GET PARTS

When a URL has been parsed or parts have been set, you can extract those pieces from the handle at any time.

```
rc = curl_url_get(h, CURLUPART_FRAGMENT, &fragment, 0);
rc = curl_url_get(h, CURLUPART_HOST, &host, 0);
rc = curl_url_get(h, CURLUPART_PASSWORD, &password, 0);
rc = curl_url_get(h, CURLUPART_PATH, &path, 0);
rc = curl_url_get(h, CURLUPART_PORT, &port, 0);
rc = curl_url_get(h, CURLUPART_QUERY, &query, 0);
rc = curl_url_get(h, CURLUPART_SCHEME, &scheme, 0);
rc = curl_url_get(h, CURLUPART_USER, &user, 0);
rc = curl_url_get(h, CURLUPART_ZONEID, &zoneid, 0);
```

Extracted parts are not URL decoded unless the user also asks for it with the *CURLU_URLDECODE* flag set in the fourth bitmask argument.

Remember to free the returned string with *curl_free(3)* when you are done with it!

## SET PARTS

A user set individual URL parts, either after having parsed a full URL or instead of parsing such.

```
rc = curl_url_set(urlp, CURLUPART_FRAGMENT, "anchor", 0);
rc = curl_url_set(urlp, CURLUPART_HOST, "www.example.com", 0);
rc = curl_url_set(urlp, CURLUPART_PASSWORD, "doe", 0);
rc = curl_url_set(urlp, CURLUPART_PATH, "/index.html", 0);
rc = curl_url_set(urlp, CURLUPART_PORT, "443", 0);
rc = curl_url_set(urlp, CURLUPART_QUERY, "name=john", 0);
rc = curl_url_set(urlp, CURLUPART_SCHEME, "https", 0);
rc = curl_url_set(urlp, CURLUPART_USER, "john", 0);
rc = curl_url_set(urlp, CURLUPART_ZONEID, "eth0", 0);
```

Set parts are not URL encoded unless the user asks for it with the *CURLU_URLENCODE* flag.

## CURLU_APPENDQUERY

An application can append a string to the right end of the query part with the *CURLU_APPENDQUERY* flag to *curl_url_set(3)*.

Imagine a handle that holds the URL "https://example.com/?shoes=2". An application can then add the string "hat=1" to the query part like this:

```
rc = curl_url_set(urlp, CURLUPART_QUERY, "hat=1", CURLU_APPENDQUERY);
```

It notices the lack of an ampersand (&) separator and injects one, and the handle's full URL then equals "https://example.com/?shoes=2&hat=1".

The appended string can of course also get URL encoded on add, and if asked to URL encode, the encoding process skips the '=' character. For example, append "candy=N&N" to what we already have, and URL encode it to deal with the ampersand in the data:

```
rc = curl_url_set(urlp, CURLUPART_QUERY, "candy=N&N",
                  CURLU_APPENDQUERY | CURLU_URLENCODE);
```

Now the URL looks like

```
https://example.com/?shoes=2&hat=1&candy=N%26N
```

## AVAILABILITY

The URL API was introduced in libcurl 7.62.0.

A URL with a literal IPv6 address can be parsed even when IPv6 support is not enabled.

## SEE ALSO

**curl_url**(3), **curl_url_cleanup**(3), **curl_url_dup**(3), **curl_url_get**(3), **curl_url_set**(3), **curl_url_strerror**(3), **CURLOPT_URL**(3)