

---

## unfurl

Pull out bits of URLs provided on `stdin`

### Install

If you have Go installed and configured:

```
1 ►  
2 go install github.com/tomnomnom/unfurl@latest
```

Otherwise download the latest binary for your platform, extract it and move it to somewhere in your `$PATH` (e.g. `/usr/bin/`):

```
1 ►  
2 wget https://github.com/tomnomnom/unfurl/releases/download/v0.0.1/  
   unfurl-linux-amd64-0.0.1.tgz►  
3 tar xzf unfurl-linux-amd64-0.0.1.tgz►  
4 sudo mv unfurl /usr/bin/
```

### Usage

unfurl works with URLs provided on `stdin`; they might come from a file like this one:

```
1 ►  
2 cat urls.txt  
3 https://sub.example.com/users?id=123&name=Sam  
4 https://sub.example.com/orgs?org=ExCo#about  
5 http://example.net/about#contact
```

### Domains

You can extract the domains from the URLs with the `domains` mode:

```
1 ►  
2 cat urls.txt | unfurl domains  
3 sub.example.com  
4 sub.example.com  
5 example.net
```

If you don't want to output duplicate values you can use the `-u` or `--unique` flag:

```
1 ►  
2 cat urls.txt | unfurl --unique domains
```

---

```
3 sub.example.com
4 example.net
```

The `-u/--unique` flag works for all modes.

## Apex Domains

You can extract the apex part of the domain (e.g. the `example.com` in `http://sub.example.com`) using the `apexes` mode:

```
1 ►
2 cat urls.txt | unfurl -u apexes
3 example.com
4 example.net
```

## Paths

```
1 ►
2 cat urls.txt | unfurl paths
3 /users
4 /orgs
5 /about
```

## Query String Keys

```
1 ►
2 cat urls.txt | unfurl keys
3 id
4 name
5 org
```

## Query String Values

```
1 ►
2 cat urls.txt | unfurl values
3 123
4 Sam
5 ExCo
```

---

## Query String Key/Value Pairs

```
1 ►
2 cat urls.txt | unfurl keypairs
3 id=123
4 name=Sam
5 org=ExCo
```

## JSON

```
1 ►
2 cat urls.txt | unfurl json
3 {"scheme":"https","opaque":"","user":"","host":"sub.example.com","path":
  :"/users","raw_path":"","raw_query":"id=123\u0026name=Sam","fragment":
  "":"","parameters":[{"key":"id","value":"123"}, {"key":"name","value":
  "Sam"}],"url":"https://sub.example.com/users?id=123\u0026name=Sam","
  domain":"sub.example.com","subdomain":"sub","root":"example","tld":
  "com","apex":"example.com","port":"","extension":""}
4 {"scheme":"https","opaque":"","user":"","host":"sub.example.com","path":
  :"/orgs","raw_path":"","raw_query":"org=ExCo","fragment":"about","
  parameters":[{"key":"org","value":"ExCo"}],"url":"https://sub.
  example.com/orgs?org=ExCo#about","domain":"sub.example.com","
  subdomain":"sub","root":"example","tld":"com","apex":"example.com","
  port":"","extension":""}
5 {"scheme":"http","opaque":"","user":"","host":"example.net","path":"/
  about","raw_path":"","raw_query":"","fragment":"contact","parameters
  ":null,"url":"http://example.net/about#contact","domain":"example.
  net","subdomain":"","root":"example","tld":"net","apex":"example.net
  ","port":"","extension":""}
```

## Custom Formats

You can use the `format` mode to specify a custom output format:

```
1 ►
2 cat urls.txt | unfurl format %d%p
3 sub.example.com/users
4 sub.example.com/orgs
5 example.net/about
```

The available format directives are:

```
1 %% A literal percent character
2 %s The request scheme (e.g. https)
3 %u The user info (e.g. user:pass)
4 %d The domain (e.g. sub.example.com)
```

---

```
5 %S The subdomain (e.g. sub)
6 %r The root of domain (e.g. example)
7 %t The TLD (e.g. com)
8 %P The port (e.g. 8080)
9 %p The path (e.g. /users)
10 %e The path's file extension (e.g. jpg, html)
11 %q The raw query string (e.g. a=1&b=2)
12 %f The page fragment (e.g. page-section)
13 %@ Inserts an @ if user info is specified
14 %: Inserts a colon if a port is specified
15 %? Inserts a question mark if a query string exists
16 %# Inserts a hash if a fragment exists
17 %a Authority (alias for %u%@%d%:%P)
```

Any characters that don't match a format directive remain untouched:

```
1 ►
2 cat urls.txt | unfurl -u format "%d (%s)"
3 sub.example.com (https)
4 example.net (http)
```

Note that if a URL does not include the data requested, there will be no output for that URL:

```
1 ►
2 echo http://example.com | unfurl format "%P"►
3 echo http://example.com:8080 | unfurl format "%P"
4 8080
```

## Help

```
1 ►
2 unfurl -h
3 Format URLs provided on stdin
4
5 Usage:
6 unfurl [OPTIONS] [MODE] [FORMATSTRING]
7
8 Options:
9 -u, --unique    Only output unique values
10 -v, --verbose   Verbose mode (output URL parse errors)
11
12 Modes:
13 keys           Keys from the query string (one per line)
14 values         Values from the query string (one per line)
15 keypairs       Key=value pairs from the query string (one per line)
16 domains        The hostname (e.g. sub.example.com)
17 paths          The request path (e.g. /users)
18 apexes         The apex domain (e.g. example.com from sub.example.com)
19 json           JSON encoded url/format objects
```

---

```
20  format    Specify a custom format (see below)
21
22  Format Directives:
23  %%        A literal percent character
24  %s        The request scheme (e.g. https)
25  %u        The user info (e.g. user:pass)
26  %d        The domain (e.g. sub.example.com)
27  %S        The subdomain (e.g. sub)
28  %r        The root of domain (e.g. example)
29  %t        The TLD (e.g. com)
30  %P        The port (e.g. 8080)
31  %p        The path (e.g. /users)
32  %e        The path's file extension (e.g. jpg, html)
33  %q        The raw query string (e.g. a=1&b=2)
34  %f        The page fragment (e.g. page-section)
35  %@        Inserts an @ if user info is specified
36  %:        Inserts a colon if a port is specified
37  %?        Inserts a question mark if a query string exists
38  %#        Inserts a hash if a fragment exists
39  %a        Authority (alias for %u%@%d%:%P)
40
41  Examples:
42  cat urls.txt | unfurl keys
43  cat urls.txt | unfurl format %s://%d%p?%q
```