



Web Crawling (1)

Robots/Web Crawlers/Spiders

- approx half of all web traffic is not human
- Robots visit websites for variety fo reasons:
 - indexing websites for searching or archiving purposes
 - e.g. to build a search engine you need to build a database of websites and note down what they contain
 - custom built robots carrying out specific tasks for their owners/programmers, e.g. newsreader like tools to give you a customised newsfeed from numerous different websites
 - probing sites for security vulnerabilities — to exploit usually
 - pretending to be humans for nefarious purposes e.g. advertise something on social media
 - generating false traffic so website owners can scam advertising networks

wget

Basic wget usage:

```
wget <url?
```

- uses this to download any file from the command line the contents of the url you `wget`

Downloading a webpage using `wget` won't make it appear exactly like it does in your browser. This is because the resources being referenced on the webpage

dont **exist locally on your filesystem**

- e.g. the specific style sheet may only exist on the server and not locally
- you could simply make a `wget` request for that style sheet and download it locally, however this will be be very efficient if the webpage includes many different resources you do not have access to

You can get around this by including a flag which lets wget know to **download all the requisites** to accurately render a page

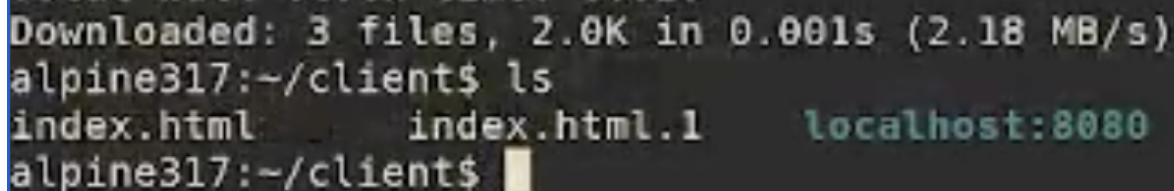
Page Requisites

```
wget -p <url>
```

- the `wget -p` will download all the requisites
 - wont work on **every site** but will work on most

Using this, `wget` will create a folder in the local directory with the name of the `<url>` you requested:

```
wget -p localhost:8080/index.html
```



A terminal window showing the output of a wget command. The first line says "Downloaded: 3 files, 2.0K in 0.001s (2.18 MB/s)". The second line shows the prompt "alpine317:~/client\$ ls". The third line shows the output of the ls command: "index.html index.html.1 localhost:8080". The fourth line shows the prompt "alpine317:~/client\$ " with a cursor.

- this is helpful for organisation, and is **especially useful** if you are ever crawling pages from **multiple domains**
- wget stores the files it gets from that domain in a directory named after that domain

In that directory it will contain the page itself as well as its requirements (e.g. the style sheet)

```
alpine317:~/clients$ ls
index.html      index.html.1    localhost:8080
alpine317:~/clients$ ls localhost\:8080/
catstyle.css   index.html      robots.txt
```

Another file will also be included in this new directory, even though we didnt ask for it and it isnt mentioned anywhere in index.html: **robots.txt**

robots.txt

- this is a standard for websites to use when they are communicating with bots that may be attempting to crawl their pages
 - websites announce their crawling preferences
- whenever you use `wget` to download more than a *single file* from a site, it will first check that domain's `robots.txt` file which is always stored in the **web root** (the top level directory of the website)
- robots.txt contains **rules** which specify which sorts of *user-agents* should be allowed to access which parts of the websites
 - there may be restrictions on accessing parts of the site therefore `wget` will respect these
- robots.txt is **not a security feature** if cannot actually stop something from accessing parts
 - you need real security measures to do that such as authentication
 - its more of an honour system to enable good bots to respect the wishes of website owners

Back to the webpage

- one thing that doesn't work on the webpage you got through `wget` will be the links at least the links that are defined *relative to the webpage* won't work as we **don't have copies of them locally**
- links which are **full urls** will still work however
- you can get `wget` to rewrite the other links as full urls by using the `-k` or `--convert-links` flags

But say if you want the ability to download multiple webpages at once from a domain...

Recursive Downloading

- you can pass options to `wget` to tell it to download a specific page and then to also *recursively* download the pages on the same domain which are linked to the original page

To download a webpage and linked pages:

```
wget -r -l N <url>
```

- the `-r` means recursive download
- by default, it is constrained to only pages on the same domain as that in the webpage URL
- the `-l N` is the level or recursion to permit **(the default if omitted is 5)**
 - if you increase the level of recursion, `wget` will get all the links from the pages being downloaded and download those pages
 - this can lead to many many pages that are being downloaded

Mirroring

- to download the entire website using wget you would use the `-m` flag

```
wget -m <url>
```

- uses standard defaults (such as setting the recursion limit to **infinite**) for creating a local copy of a *full website*
 - it will continue following links until it runs out of links to follow

If you were to run this you might want to run it with another option: `-w 1`

- this creates a 1 second delay between each request which avoids annoying a server that doesn't link lots of requests per second from the same address
 - some servers may block you for spamming requests

`wget -r -l 1` example questions:

Question 1

```
wget -r -l 1 http://example.com
```

1. **r**: This option tells `wget` to operate recursively, meaning it will follow links and download content from the pages those links point to.
2. **l 1**: This option sets the recursion depth to 1. In other words, `wget` will download the initial page (the one you point it at) and any resources directly linked from that page, such as images, CSS files, and linked HTML pages, but it will not follow links on those secondary pages.

```
<!DOCTYPE html> <html>
<head>
<meta charset="UTF-8">
<link type="text/css" href="style.css" rel="stylesheet">
```

```
<title>Example Page</title> </head>
<body>
<h1>Example Page</h1>
<p>See the <a href="./about.html">about</a> page for more
about this site, or check out
the <a href="https://www.iana.org/domains/example"> IANA resour
more about example pages.</p>
</body> </html>
```

- **Initial Page (`http://example.com/index.html`):**
 - The index page itself is downloaded.
 - The CSS file linked in the header (`style.css`) is downloaded because it's directly linked from the index page.
 - The "about.html" page is also downloaded because it's directly linked from the index page.
 - Any other resources like images or scripts directly linked on the index page would also be downloaded.
- **Secondary Links from "about.html":**
 - Despite "about.html" likely containing its own links to other local or external pages, `wget` will not follow these because the recursion limit is set to 1.
- **External Links:**
 - The link to "`https://www.iana.org/domains/example`" would not be followed because `wget` by default does not download from different domains unless explicitly configured to do so (and because of the recursion limit).

Question 2

Alex is using `wget` to download resources from a website, `example.org`. The homepage (`index.html`) of `example.org` includes the following HTML:

```
<!DOCTYPE html>
<html>
```

```

<head>
  <meta charset="UTF-8">
  <link type="text/css" href="main.css" rel="stylesheet">
  <script src="scripts.js"></script>
  <title>Home Page</title>
</head>
<body>
  <h1>Welcome to Our Site</h1>
  <p>Learn more on the <a href="./info.html">Information I
    or visit our <a href="./contact.html">Contact Pa
    
  </body>
</html>

```

If Alex downloads the homepage using the command:

```
wget -r -l 1 http://example.org
```

Question: How many files will Alex retrieve, and which ones?

- 6 files total: `index.html` , `main.css` , `script.js` , `info.html` , `contact.html` , `logo.png`

Question 3

Scenario: Jessica is looking to archive a section of a blog hosted at `blog.example.net`. The section's main page (`section.html`) contains:

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="section.css">
  <title>Blog Section</title>
</head>
<body>
  <h1>Featured Articles</h1>
  <p>Check out the <a href="./article1.html">first article</a>

```

```
or explore the <a href="./article2.html">second article</a>.</p>
</body>
</html>
```

Jessica runs the command:

```
wget -r -l 1 http://blog.example.net/section.html
```

Question: What files will Jessica download, and what will not be included?

- `section.html` , `section.css` , `article1.html` , `article2.html`
- not included: any resources or links embedded within article1 or article2

Question 4

Scenario: Emily is tasked with archiving a small section of a website, `educational.example.com` . The main page of this section (`index.html`) contains:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="style.css">
<script src="scripts.js"></script>
<title>Educational Resources</title>
</head>
<body>
<h1>Welcome to Our Educational Resources</h1>
<p>Explore our <a href="./courses.html">Courses</a> or read o
ur <a href="./news.html">Latest News</a>.</p>
</body>
</html>
```

The `courses.html` page, linked from the main page, contains the following HTML:


```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="boufous.css">
<title>Courses Offered</title>
</head>
<body>
<h1>Our Courses</h1>
<p>Check out the detailed syllabus for the <a href="./syllabus.html">
  Advanced Biology</a> course.</p>
</body>
</html>
```

Emily uses the following command to download the content:

```
wget -r -l 2 http://educational.example.com
```

Question: What files will Emily download, and which files will not be retrieved?

Ethics

- crawling can be ethically different to browsing a website

Concerns

- you need to respect the limits that are set out in the `robots.txt` file
 - any crawler you are using should respect the rules set

- Established tools such as `wget` will do this as standard but if you create a custom crawler you will need to implement this behaviour
- you might need to assess if web crawling is the best way to get a copy of a webpage
- read permissions on a website don't always mean you are allowed to republish downloaded content - e.g. copyright issues
- Besides legal concerns, is republishing something ethical? — making social media sites public might infringe privacy confidence
- avoid crawling which may have security risks to a site
- some web servers impose bans on clients if they are rapidly and automatically making requests as it blocks other human clients, it is generally polite to incur a **short crawl delay** between requests

Other limits on web crawling

- you have generally unrestricted rights access

need notes from lecture ...