Advanced - Conclusion

EECS 201 Winter 2022

Submission Instructions

This assignment will be submitted as a repository on the UMich GitLab server. Create a private, blank, **README-less (uncheck that box!)** Project on it with the name/path/URL `eecs201-adv-conclusion` and add `brng` as a Reporter. The submission branch will be `index`. If this branch is not already the default initial branch, you initialize the local repo with an additional argument: `git init --initial-branch=index` if your version of Git is recent enough. Otherwise you can create a branch with this name after your first commit.

Submission Instructions

Repo directory structure:

```
/
|-- report.txt
|-- eecs201-download
```

Preface

Websites aren't magic. When you enter a URL in your web browser, like `https://www.eecs.umich.edu/courses/eecs201/wn20` your browser looks up what to connect to via a domain name of `www.eecs.umich.edu`, which is mapped to an IP address to find the exact computer to connect to via domain name resolution, and makes a connection to that computer. Your browser then utilizes the HTTP protocol (or rather, HTTPS, as indicated by `https://`) to tell the server that it wants a resource located at `/courses/eecs201/wn2022/`, and the server obliges by sending you a webpage (e.g. an HTML file) for your browser to render and display to you. It not need be a webpage, it could be a PDF file, an image file, a tarball, or any manner of file.

At their simplest, websites are just file servers you communicate with via the HTTP(S) where you provide a URL and the server responds by providing you the requested file in that website's directory hierarchy (or a 404 webpage to indicate that it couldn't find a file of that name). More complex websites could decide to do something else with the URL and request, such as handle requests to upload data and store it in a database, dynamically generate webpages and send them over instead of a static HTML file, give you some raw data such as that for the weather, and more.

If you have ever made a website, you might have come across something along the likes of `index.html` and the like. What happens when you try to "download" a directory when you navigate to a website's "home" page just by entering in a URL like `https://www.coolwebsite.com`, where you haven't specified a particular file, thus dumping you at the root of the website's space on its server? This is where `index.html` *et al.* come into play: they are what is served when you request a directory. A website, for instance, could use it to serve a home page (which I do with `https://www.eecs.umich.edu/courses/eecs201/wn2022`, `courses/eecs201/wn2022/` is a directory on the EECS webserver), or could use it to serve a webpage that is an *index* (i.e. a listing) of the files in that directory. Oftentimes, webserver software will by default serve this directory listing when a user hasn't provided an explicit `index.html` *et al.*: this is known as an "open directory" which could possibly be an unintended security concern if there's other secret files that aren't meant to be publically listed.

Why am I telling you all this? Well let's take a look at my final challenge to you :)

1 Download the course resources (10)

In this assignment you'll automatically download all the lecture and assignment PDFs. If you navigate to https://www.eecs.umich.edu/courses/eecs201/wn2022/files/assignments/ and https://www.eecs.umich.edu/courses/eecs201/wn2022/files/lectures/, you'll see that I've set up an index page for where these PDFs are. Your job is to write a Bash or Python 3 script that will take advantage of these index pages to download all the files that they link to.

Formally, write a script named `eecs201-download` (no extension!) that:

1. Utilizes Bash or Python 3
2. Has an appropriate shebang (`#!/bin/bash`, `#!/usr/bin/env python3`)
3. Has the the execute permission/mode set
4. Retrieves the directory listing HTML files for https://www.eecs.umich.edu/courses/eecs201/wn2022/files/assignments/ and https://www.eecs.umich.edu/courses/eecs201/wn2022/files/lectures/
5. Utilizes the directory listing HTML files to figure out what files to download (they need not necessarily be PDFs: download everything listed)
6. Downloads the files to the current directory

If you're having trouble getting started, remember that in several assignments we used `wget` or `curl` to make HTTP requests to download the assignment tarballs; the same thing will work here (your URL will probably need to have a `/` at the end to get the index file). HTML files are plain-text, so you can do some processing on it to extract the needed information.

Notes:

1. **Do not commit the PDFs!**
2. If you are using Python 3, you may only use the standard library and whatever other system installed packages there are on the course server. (Yes, `requests` is already installed)
3. If you're writing a Bash script, `cut` and `awk` might be helpful
4. This assignment should work on the course server.
5. Some aspects of this might be familiar if you did a certain Advanced Project.
6. Don't think you can get away with hard-coding your result. The index file is free to update whenever it wants to: there might be secret files added and some files might be removed after the deadline ;)
7. The files will be directly located in the directory of that given index e.g. `https://www.eecs.umich.edu/courses/eecs201/`
8. If you are unfamiliar with HTML, an `<a>` tag is able to create a clickable hyperlink to a resource. The `href` field specifies where to find the resouce while the text between the opening tag (`<a>`) and closing tag (`</a>`) is the visible text. Both may be helpful for you in writing your script. Note how the index file's `href`s only specify the path part of the URL and don't include the domain name and protocol (the `https://www.eecs.umich.edu`).

2 Conclusion

1. Commit your `eecs201-download` script.
2. Create a file called `report.txt`.
3. On the first line provide an integer time in minutes of how long it took for you to complete this assignment.
4. On the second line and beyond, write down what you learned while doing this assignment. If you already knew how to do all of this, put down "N/A".
5. Add and commit this `report.txt` file. Push your commits to a branch called `index`.