# CS 432/532: Web Science

Spring 2017

## Assignment 1

Michael Micros

## Honor Pledge

I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagIiarism. I am aware that as a member of the academic community it is my responsibility to turn in all suspected violations of the Honor Code. I will report to a hearing if summoned.

Signed _____

January 26, 2017

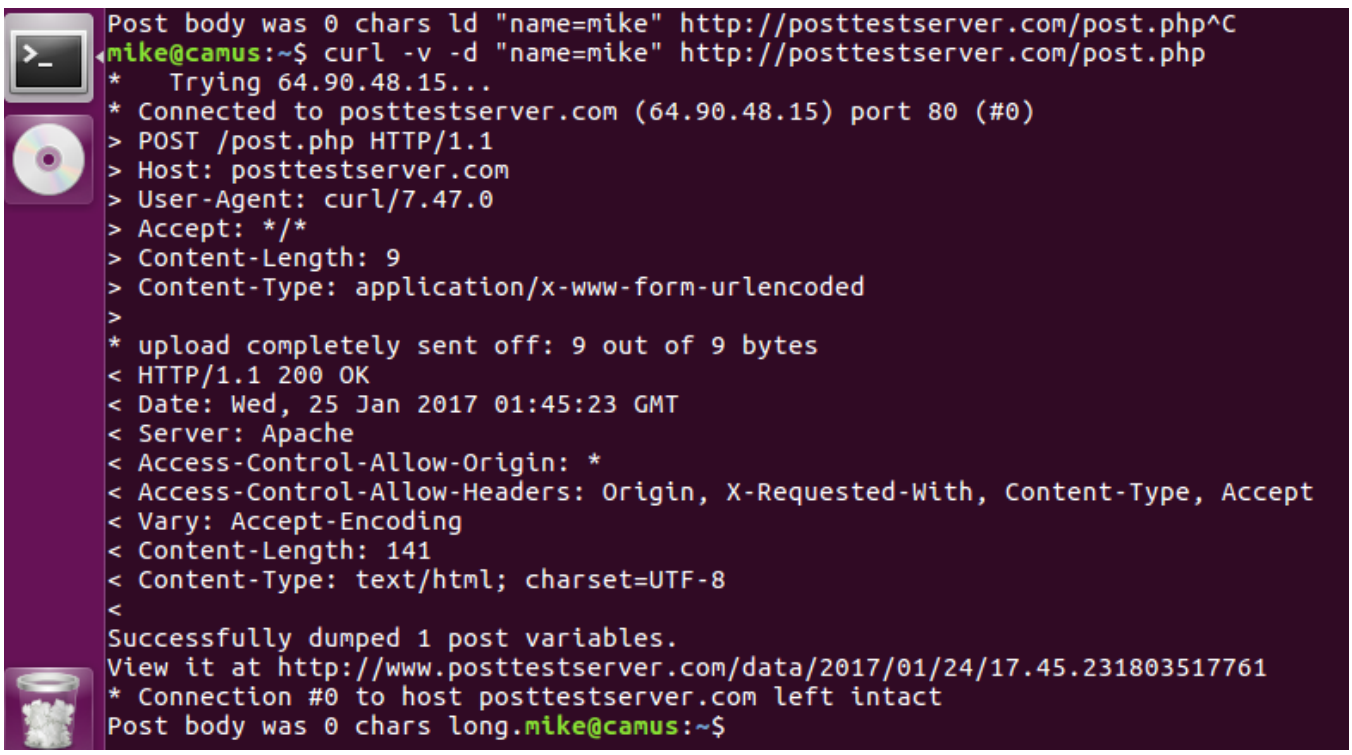
# Problem 1

## Question

1. Demonstrate that you know how to use "curl" well enough to correctly POST data to a form. Show that the HTML response that is returned is "correct". That is, the server should take the arguments you POSTed and build a response accordingly. Save the HTML response to a file and then view that file in a browser and take a screen shot.

## Answer

In order to answer this question, all that was necessary was to read through the curl command manual, which is found by typing "man curl" into the command line.
The exact command used was "curl -v -d" as can be seen in Fig. 1.
Option "-v" makes curl verbose during the operation.
Option "-d <data>" sends the data as a POST request to the HTTP server.
Finally, the form to which the data was POSTed was found from a website that allows the user to POST and inspect the generated response as seen in Fig. 2.



```
Post body was 0 chars ld "name=mike" http://posttestserver.com/post.php^C
mike@camus:~$ curl -v -d "name=mike" http://posttestserver.com/post.php
*   Trying 64.90.48.15...
* Connected to posttestserver.com (64.90.48.15) port 80 (#0)
> POST /post.php HTTP/1.1
> Host: posttestserver.com
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 9
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 9 out of 9 bytes
< HTTP/1.1 200 OK
< Date: Wed, 25 Jan 2017 01:45:23 GMT
< Server: Apache
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
< Vary: Accept-Encoding
< Content-Length: 141
< Content-Type: text/html; charset=UTF-8
<
Successfully dumped 1 post variables.
View it at http://www.posttestserver.com/data/2017/01/24/17.45.231803517761
* Connection #0 to host posttestserver.com left intact
Post body was 0 chars long.mike@camus:~$
```

Figure 1: curl command in terminal

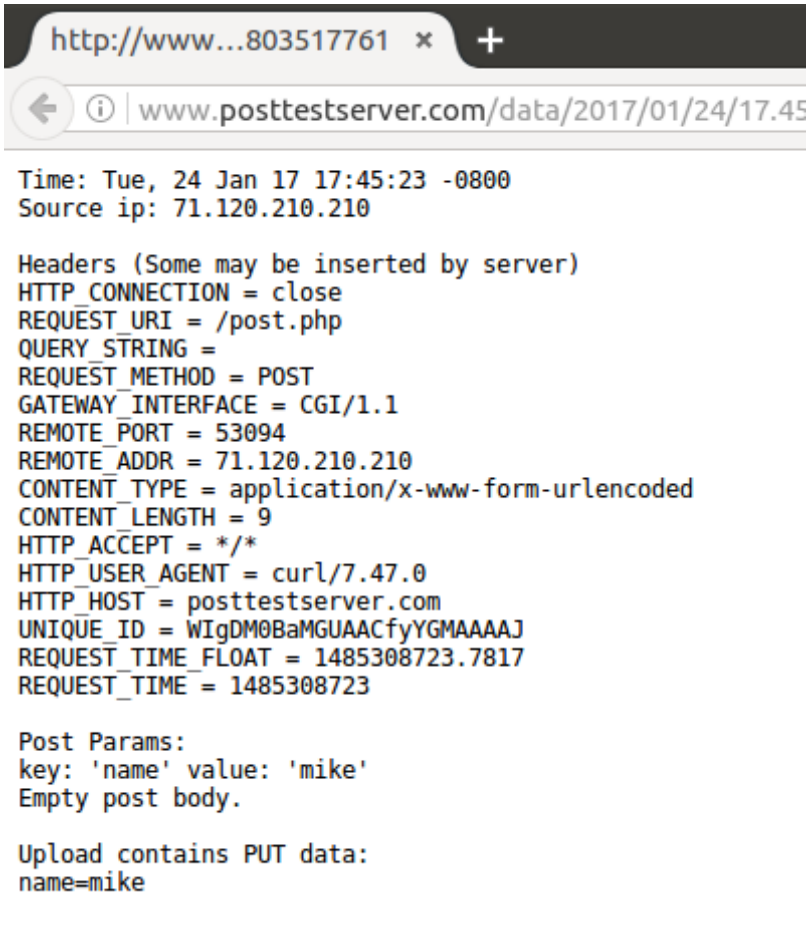



Figure 2: HTML responsel

# Problem 2

## Question

```
2.  Write a Python program that:
  1. takes as a command line argument a web page
  2. extracts all the links from the page
  3. lists all the links that result in PDF files, and prints out
     the bytes for each of the links.  (note: be sure to follow
     all the redirects until the link terminates with a "200 OK".)
  4. show that the program works on 3 different URIs, one of which
     needs to be:
     http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html
```

## Answer

1. In order  for our Python program to accept a command line argument, 'sys' must be imported. This allows us to access the arguments that are entered at the command line. "argv[0]" is the command that executes the python script and "argv[1]" is the web page that the user selected.

2. After the desired web page was opened using "urllib.request.openurl() ", Beautifulsoup was used to parse all the links on the page using the instructions:

```
>>>  soup = BeautifulSoup(Html,"html.parser")
>>> for link in soup.find_all('a'):
```

3. In order to detect the content type of each link, a new request was made for each link obtained in the previous part. Using urllib we are able to search the info of the response to check if the response code was "200 OK" and whether the "Content-type" was "application/pdf". I specifically searched the content type for anyninstance of the string "pdf" because there are cases where the content is of type PDF but the notation is slightly different than the standard. The instruction that does all this is:

```
>>>if('pdf' in linkRes.info()['Content-Type'] and (linkRes.code== 200) ):
```

4.The results of the program can be seen in Figure 4 and 5.

```python
#!/usr/bin/env python

from bs4 import BeautifulSoup
import urllib.request
import sys

url=sys.argv[1]
#url = "http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html"
req=urllib.request.Request(url)

with urllib.request.urlopen(req) as res:
        Html = res.read()
        soup = BeautifulSoup(Html,"html.parser")
        for link in soup.find_all('a'):
                linkReq = urllib.request.Request(link.get('href'))
                with urllib.request.urlopen(linkReq) as linkRes:
                        if('pdf' in linkRes.info()['Content-Type'] ):
                                print("First URI : " + link.get('href'))
                                print("Last URI  : {}".format(linkRes.geturl()))
                                print("Bytes: " + linkRes.info()['Content-length'] +"\n")
```

Figure 3: p2.py (Located in Problem2 directory)

```
mike@camus:~/python_Programs$ python3 test2.py http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdf
s.html
First URI : http://www.cs.odu.edu/~mln/pubs/ht-2015/hypertext-2015-temporal-violations.pdf
Last URI  : http://www.cs.odu.edu/~mln/pubs/ht-2015/hypertext-2015-temporal-violations.pdf
Bytes: 2184076

First URI : http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-annotations.pdf
Last URI  : http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-annotations.pdf
Bytes: 622981

First URI : http://arxiv.org/pdf/1512.06195
Last URI  : https://arxiv.org/pdf/1512.06195.pdf
Bytes: 1748961

First URI : http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-off-topic.pdf
Last URI  : http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-off-topic.pdf
Bytes: 4308768

First URI : http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-stories.pdf
Last URI  : http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-stories.pdf
Bytes: 1274604

First URI : http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-profiling.pdf
Last URI  : http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-profiling.pdf
Bytes: 639001

First URI : http://bit.ly/1ZDatNK
Last URI  : http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-temporal-intention.pdf
Bytes: 720476

First URI : http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-mink.pdf
Last URI  : http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-mink.pdf
Bytes: 1254605

First URI : http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-arabic-sites.pdf
Last URI  : http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-arabic-sites.pdf
Bytes: 709420

First URI : http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-dictionary.pdf
Last URI  : http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-dictionary.pdf
Bytes: 2350603
```

Figure 4: p2.py (Result for http://www.cs.odu.edu/ mln/teaching/cs532-s17/test/pdfs.html )

# Problem 3

## Question

```
3.  Consider the "bow-tie" graph in the Broder et al. paper (fig 9):
    http://www9.org/w9cdrom/160/160.html

    Now consider the following graph:

    A --> B
    B --> C
    C --> D
    C --> A
    C --> G
    E --> F
    G --> C
    G --> H
    I --> H
    I --> K
    L --> D
    M --> A
    M --> N
    N --> D
```

```
mike@camus:~/python_Programs$ python3 p2.py https://sites.wp.odu.edu/ASPEN/publications/
First URI : http://ww2.odu.edu/~dkrusien/papers/JNE2015.pdf
Last URI  : http://ww2.odu.edu/~dkrusien/papers/JNE2015.pdf
Bytes: 1112253

First URI : http://ww2.odu.edu/~dkrusien/papers/BMEL2015.pdf
Last URI  : http://ww2.odu.edu/~dkrusien/papers/BMEL2015.pdf
Bytes: 393921

First URI : http://ww2.odu.edu/~dkrusien/papers/JNE2014b.pdf
Last URI  : http://ww2.odu.edu/~dkrusien/papers/JNE2014b.pdf
Bytes: 2680216

First URI : http://ww2.odu.edu/~dkrusien/papers/JNE2014a.pdf
Last URI  : http://ww2.odu.edu/~dkrusien/papers/JNE2014a.pdf
Bytes: 1810376

First URI : http://ww2.odu.edu/~dkrusien/papers/TAES2013.pdf
Last URI  : http://ww2.odu.edu/~dkrusien/papers/TAES2013.pdf
Bytes: 18212927

First URI : http://ww2.odu.edu/~dkrusien/papers/GRSL2013.pdf
Last URI  : http://ww2.odu.edu/~dkrusien/papers/GRSL2013.pdf
Bytes: 776691

First URI : http://ww2.odu.edu/~dkrusien/papers/MCP2012.pdf
Last URI  : http://ww2.odu.edu/~dkrusien/papers/MCP2012.pdf
Bytes: 862778

First URI : http://ww2.odu.edu/~dkrusien/papers/BRB2012.pdf
Last URI  : http://ww2.odu.edu/~dkrusien/papers/BRB2012.pdf
Bytes: 541898
```

Figure 5: p2.py (Result for https://sites.wp.odu.edu/ASPEN/publications/)

```
O --> A
P --> G

For the above graph, give the values for:

IN:
SCC:
OUT:
Tendrils:
Tubes:
Disconnected:
```

## Answer

After mapping out the graph that was given (the Figure below), it becomes evident which are the inputs, outputs and strongly connected components. The components of the graph that are not pointed(linked) to by other componentss and have at least one connection to the SCC are INPUTS . The components that do not point to anything but are pointed(linked) to by at least one component in the SCC are OUTPUTS. The components that are all interconnected form the SCC area.The nodes that do not connect to any of the other components are DISCONNECTED. Finally the nodes that connect only to the IN or OUT of the graph without connecting to the SCC are the TENDRILS.

```
IN:  M,O,P
```

```
SCC:  A,B,C,G
OUT:  H,D
Tendrils: I,K,L
Tubes: N
Disconnected: E,F
```

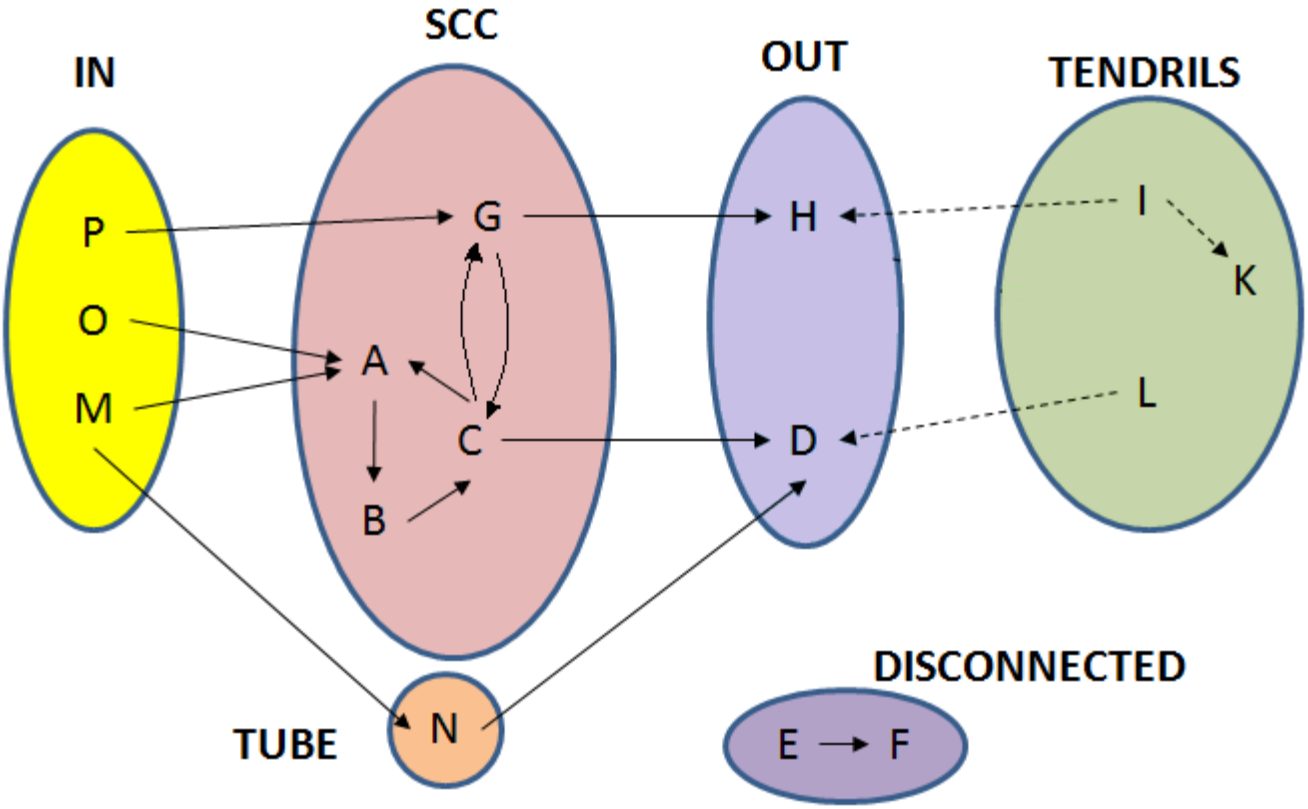



Figure 6: Bow Tie Graph