

CS 532: Assignment 1

Dinesh Kumar Paladhi

Spring 2016

Contents

1	Problem 1	2
1.1	Solution	2
1.2	Results	3
2	Problem 2	5
2.1	Solution	5
2.2	Results	5
2.3	Code Listing	7
3	Problem 3	8
3.1	Solution	8

1 Problem 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.

1.1 Solution

1. Started with understanding what curl does from the secure shell by using curl -help command.
2. Got various options which are used in curl. Later,Referred few sites in order to get in touch with curl.
3. My next step was to get the web forms which use POST method but It was hard to find them. Later, I found the link in the goggle groups and started using it.
4. I selected the following web page to post my data.

```
http://www.cs.tut.fi/cgi-bin/run/~jkorpela/echo.cgi
```

5. The curl command i used to post data to a form is

```
Curl --data "comments=Good to see you"  
http://www.cs.tut.fi/cgi-bin/run/~jkorpela/echo.cgi -o output.html
```

6. In the beginning i did not use -o output.html and so i got the html code as output on my command prompt.
7. curl -o output.html command creates a html file in the Z drive.

1.2 Results

1. Screen shot of the web page with the form to which I posted data is shown below.

The screenshot shows a web browser window with multiple tabs. The active tab is titled "Testing HTML forms" and has the URL "https://www.cs.tut.fi/~jkorpela/forms/testing.html". The page content is as follows:

problem by checking whether the script gets the data correctly

- when testing browsers--notice that there are some browser-dependent features in sending data from forms.

The scripts mentioned here work both for METHOD="GET" and for METHOD="POST".

My simple sendback script

My simple sendback script (written in [Perl](#) using the [CGI.pm](#) library) echoes the data it gets, with field names in one column and and field values in another column, in monospaced font. However, the data is **decoded** into plain text.

The script can handle forms with INPUT TYPE="FILE" fields (i.e. forms with ENCTYPE="multipart/form-data") too.

To use the script, simply take your FORM element and change the ACTION attribute to ACTION="http://www.cs.tut.fi/cgi-bin/run/~jkorpela/echo.cgi"

Example:

```
<FORM ACTION="http://www.cs.tut.fi/cgi-bin/run/~jkorpela/echo.cgi"
METHOD="POST">
<P>
Type something:<BR>
<TEXTAREA ROWS=5 COLS=72 NAME=Comments>
This is
some text
in several lines.
</TEXTAREA>
<p>
<INPUT TYPE="checkbox" NAME="box" VALUE="yes">Check me!
<p>
<INPUT TYPE="HIDDEN" NAME="hidden field" VALUE="something">
<INPUT TYPE="SUBMIT" VALUE="Send">
</FORM>
```

Try it:

Type something:
Good to see you

Check me!

Figure 1: Form in the Web page

2. The response I got after implementing the curl command is shown below

```
atria:~> curl --data "comments=Good to see you" http://www.cs.tut.fi/cgi-bin/run  
/~jkorpela/echo.cgi -o output.html  
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current  
          Dload  Upload   Total   Spent    Left  Speed  
100  425    0  401  100    24     300     17  0:00:01  0:00:01  ---:--  300
```

Figure 2: Response for the curl command

3. Screen shot of the html file in the browser is shown below.

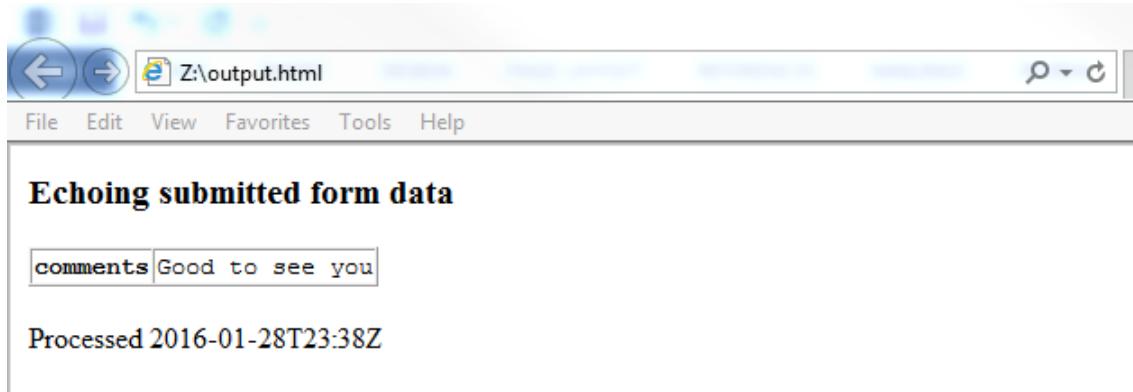


Figure 3: Response from the browser

2 Problem 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-s16/test/pdfs.html/](http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html)

2.1 Solution

1. First I tried to write the program and gather all the links from one particular link like www.cs.odu.edu.
2. For doing this I imported the required libraries.
3. Extracted the code using BeautifulSoup library.
4. Read the entire data from the above url using urllib2 and BeautifulSoup libraries.
5. Got all the data within the anchor tag using BeautifulSoup function.
6. Wrote a sample function which checks for the url and returns false if it is not.
7. Now I ran a for loop where it stores the headers of each link and then also checks whether the content-type in the headers is pdf or not. If it is pdf then I am printing the url, content-length and the status code.
8. Similar process is carried on with 2 other links and the results are shown below.

2.2 Results

1. To get the sample output 1 the following command should be executed

```
python prob2.py http://www.cs.odu.edu
```

```
atria:~/Webscience/cs532-s16/Assignment 1/Latex> python prob2.py http://www.cs.odu.edu
link    Content length      Response Code
http://www.cs.odu.edu/studentappointmentinfo.pdf      636560      200
http://www.cs.odu.edu/StrategicPlan0515_2010.pdf      909323      200
http://www.cs.odu.edu/files/cs_systems_services.pdf      412031      200
http://www.cs.odu.edu/files/csintroductioninfo.pdf      564602      200
atria:~/Webscience/cs532-s16/Assignment 1/Latex> █
```

Figure 4: Output 1

2. To get the sample output 2 the following command should be executed

```
python prob2.py http://www.cs.odu.edu/~mln/teaching/cs532-s16/test/pdfs.html
```

```
atria:~/Webscience/cs532-s16/Assignment 1/Latex> python prob2.py http://www.cs.odu.edu/~mln/teaching  
link Content length Response Code  
http://www.cs.odu.edu/~mln/pubs/ht-2015/hypertext-2015-temporal-violations.pdf 2184076 200  
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-annotations.pdf 622981 200  
http://arxiv.org/pdf/1512.06195 1748961 200  
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-off-topic.pdf 4308768 200  
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-stories.pdf 1274604 200  
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-profiling.pdf 639001 200  
http://bit.ly/1ZDatNK 720476 200  
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-mink.pdf 1254605 200  
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-arabic-sites.pdf 709420 200  
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-dictionary.pdf 2350603 200  
atria:~/Webscience/cs532-s16/Assignment 1/Latex> █
```

Figure 5: Output 2

3. To get the sample output 3 the following command should be executed

```
python prob2.py http://www.oracle.com/index.html
```

```
atria:~/Webscience/cs532-s16/Assignment 1/Latex> python prob2.py http://www.oracle.com/index.html  
link Content length Response Code  
http://www.oracle.com/us/industries/oracle-hospitality-brochure-2870500.pdf 125037 200  
█
```

Figure 6: Output 3

2.3 Code Listing

```
import urllib2
import requests
import sys
from bs4 import BeautifulSoup

def valid_url(url):    #Function to eliminate href's which are not url's
    try:
        urllib2.urlopen(url)
        return True
    except Exception, e:
        return False

url = urllib2.urlopen(sys.argv[1]).read()    #Reads the entire data from the url
                                                given in the command prompt
soup = BeautifulSoup(url)
all_links= soup.findall('a',href=True) # getting only anchor tag data
print "link","--","Content_length","--","Response_Code"  # Supportive text
for line in soup.findall('a'):
    hrefLink = line.get('href')
    if valid_url(hrefLink) : #Calling the function defined above
        response=urllib2.urlopen(hrefLink)
        link_header=response.headers
        type=link_header["Content-type"]
        if type=="application/pdf":  #Checking whether the
            file is pdf or not
            print hrefLink,"--", link_header["content-
length"],"--",response.getcode()
```

3 Problem 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 --> J
I --> K
J --> D
L --> D
M --> A
M --> N
N --> D
O --> A
P --> G
```

For the above graph, give the values for:

IN:

SCC:

OUT:

Tendrils:

Tubes:

Disconnected:

3.1 Solution

Model graph is drawn based on the bow-tie concept including all the node points stated above.

IN: M,O,P
SCC: A,B,C,G,
OUT: D,H
In Tendril: None
Out Tendril : L,K,I,J
Tubes: N
Disconnected: E,F

Explanations for each node have been provided below.

IN: O, M, P

According to the definition of IN, any node in IN can be connected to any other node in IN or can be connected to any node in SCC but not with any other nodes. M,O,P falls into this category they are connected to A,G nodes in SCC.

SCC: A, B, C, G

Strongly Connected Component consists of group of nodes which are connected to each other directly or indirectly and these nodes can be connected from IN and connected to OUT. A,B,C,G come under this category.

OUT: D, H

Nodes which exit from SCC and are not connected back to SCC come under OUT. D and H show these characteristics so they come under OUT.

Tendrils: I, J, K, L

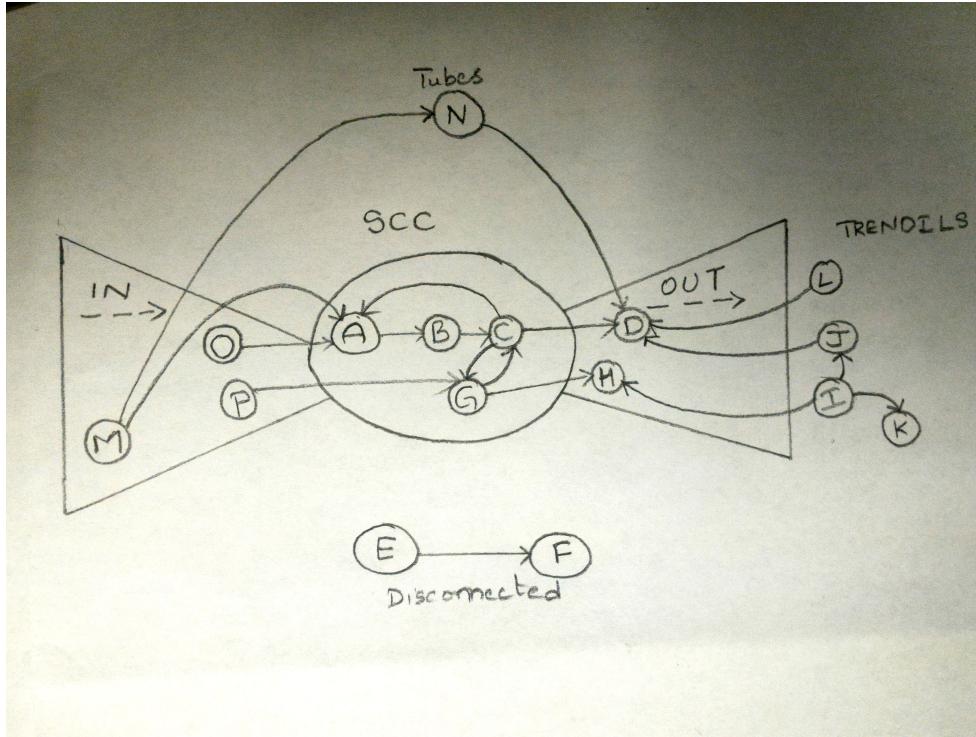


Figure 7: Drawing of the example graph

The nodes which cannot be directly connected to the nodes in SCC. I, J, K, L come under *Tendrils*.

Tubes: N

Nodes which pass from IN to OUT without any interaction with SCC are called *tubes*. N is a tube linking from M to D .

Disconnected: E, F

The Disconnected components link to no one in the graph, and stand alone. They are not defined explicitly by Broder, et. al, but their meaning is implied within the paper.

Bibliography

- [1] How to fetch Internet Resources Using urllib2,
<https://docs.python.org/2/howto/urllib2.html>.
- [2] Curl,
<http://curl.haxx.se/docs/manpage.html>.
- [3] BeautifulSoup Documentation,
<http://www.crummy.com/software/BeautifulSoup/bs4/doc/>.
- [4] Stack Overflow answering questions on Content-type,
<http://stackoverflow.com/questions/2143674/how-do-i-get-a-content-type-of-a-file-in-python-with-url>.
- [5] Graph Strdture in the Web,
<http://www9.org/w9cdrom/160/160.html> .