Homework 2 (150 points)
cs3157 – Advanced Programming
Prof. Shlomo Hershkop
Dept of Computer Science
Columbia University
Spring 2007

Out: Mar 26 2007
Due: Apr 15 2007 – 11pm

Objective:
1. Practice with c coding/compiling/debugging
2. program an original idea
3. Have fun!

Background
Most of you are familiar with the standard web browser. This is simply a program which allows one to enter a URL and view a webpage rendered in some form. To mark a specific webpage to visit later, most browsers allow the user to create ‘links’ to revisit specific web pages. The problem is that the interface to manipulate and view links is usually stored as a text only tree hierarchy, which quickly grows over time, making it completely useless to find a specific webpage if you don’t remember anything about the page you are looking for. Example: imagine you remember some webpage you bookmarked having a specific colored circle at the top of the page, but don’t remember which one. With 200 bookmarked links, you potentially would need to surf to over 200 pages to view the specific example (if it hasn’t been replaced).

Enter HOMEWORK2!!

Super links will solve all this and many more problems. This is a website which allows the user to import links, manage links, and view snapshots of links over the web. Each user has their own log in section to allow privacy on their links.

Sounds easy, I hope you at least have some fun and get creative on this homework ??

Programming Part I:
You will be using your c skills to get everything working correctly. You will need to master some other programs and skills outside of pure c in order to pull this off on the cs network.
Please see us if you run into problems along the way….don’t just sit there and try all combinations…. 
Step 1: Getting Webpage snapshots.

This will be the hardest non programming part of the assignment. Using a program called WebTHUMB you can ask it really nicely to create a snapshot of a webpage for you. Unfortunately it will try to crash as much as possible, so you need to understand what is happening on the system in order to get it working correctly. Once it works, you should find it easy to use. Lucky for you it’s been written in perl!

One of the ideas in the course, is to learn how to read error messages and how to get systems to work together….and what can go wrong. This step will be an excellent example of all this 😊

Webthumb is a perl script which ties together a few different linux programs to generate snapshots of webpages…its available from http://www.boutell.com/webthumb/

To test the basic version of your program, please choose a completely random (and most important - appropriate) website and generate a snapshot picture as a 400 by 400 jpg named step2.jpg.

Note: if you are not in the clic lab, a hint: use a vnc session to access a linux machine (and be sure to delete the .webthumb directory if you have problems). This will not work over putty.

Internally, the webthumb script tries to be smart, by only loading mozilla once into memory, and using the .webthumb directory to keep track of this. This is great assuming you are on the local system and the only user….clic can potentially have many users and you could be running this remotely. In case something goes wrong, make sure you don’t have a webthumb process running in the background when you retry the program. You can check it with the ‘ps’ command (ps –au username) (replace it with your own username). (kill -9 processnumber) will kill the running process of webthumb. You might need to kill an instance of ‘xvfb’ in case webthumb gets stuck on that.

This is important: you need to read through and see what the perl script does in order to get it working correctly. You will need to make the following modifications to the perl script and you will need to explain in the readme what these modifications accomplish.

1) one of the programs called by webthumb is Xvfb, pass it the –ac flag
   a. this requires modifying one of the lines in the perl script

2) the port for xvfb is specified by :number
   example :2
   change this value to something else (small) so you don’t conflict with another user on the system (if you have problems)
   a. IMPORTANT: this is changed in 2 locations in the perl script

3) BEFORE the step, make sure when you launch mozilla it will start with some default profile…that is make sure that if it bothers you about a profile, you setup a
default one so that the webthumb script doesn’t hang on that choice box for choosing a profile.

a. Example: if you were to launch mozilla and it asks you what profile to use…that asking window will freeze webthumb since it doesn’t interact with mozilla, it assumes it will work ☺

4) Pay attention to errors associated with temp files created in /tmp , you might have to manually remove some tmp lock files if things go wrong. Example: if webthumb complains that is can’t create temp file /tmp/X3-lock, that is because someone else is running webthumb on port 3, switch ports in step 2.

I am assuming at this point, you can generate a snapshot of any webpage. The entire step should be able to be done in about two hours with problems 😊

Next: Get the basic webpage going on cluster….remeber that webthumb ONLY runs on linux machines, we will be able to hook both the webpage and linux machines together using telnet in step 3

Step2: Basic webpage

In c, create a basic webpage asking the user to log in, or create an account. The user need to specify a username and password to access their link collection. Think about how to organize this.

Once the user is authenticated they should be able to view all links, delete specific links, and add a link into the system.

So it’s trivial if you already have a snapshot of a link to display it in html. For example: given a link www.cnn.com or http://www.cnn.com you can map both to www_cnn_com.jpg and use it as a way of displaying the thumbnail next to the link text.

Otherwise your program will need to generate a new snapshot file. The problem is that in the cs system, the webpages run on solaris, while webthumb works on linux. You need to have a way of sending urls to linux, generating a snapshot, and getting back the binary jpg information to display to the user.

This might take about 3-6 hours with testing and debugging.

Step3: telnet!

NOTE: IF YOU ARE HAVING TOO MANY PROBLEMS you can program this part in perl.

Program which allow other programs to talk to them do so usually through a socket. Each machine has a unique ip address which identifies it on a local network. In addition each machine can listen or connect to a ‘port’ which is a specific number between 0-65535,
which some of them are reserved for specific services. For example mail travels over port 25 etc.

Hint: on linux machines the ‘ifconfig’ command will show you information about a specific network hardware connection. (try ‘/sbin/ifconfig’ or ‘ifconfig –a’ if you have a problem getting it to show). (windows uses the ‘ipconfig’ command). The ‘netstat –a’ command will show you port information on the local system.

So a socket, is the equivalent of a phone device. If you plug in a phone into a phone line, you can respond to anyone who dials in on the line number. In the same sense, a socket is a communication device, which you can either connect to something remotely, or listen for a connection. NOTE: sockets can potentially sit on the same computer. The two types of sockets are server (listener) and client (talk talk talk).

C instructions:

Download sample server, and client c code (near the top of the page, server.c and client.c) from (http://www.cs.rpi.edu/courses/sysprog/sockets/sock.html). Read the description on sockets.

Compile the code, and then with 2 windows open, launch the server (by the way: localhost (your local machine) is usually mapped to 127.0.0.1 by convention, on the Windows OS this is done in c:\windows\system32\drivers\etc\hosts) on linux  this is done is /etc/hosts.

To test out a small program you can adopt the client and server to print out a message to each other when they connect. For example your server can print out “hi this is ???’s server” and the client can respond “hi this is ???’s client”.

Now you need to setup the client server to be able to pass a url to the linux system from the solaris system and run webthumb on the linux system and pass back the binary data.

Perl instructions:

For perl sockets programming use IO::Socket::INET;

NOTE2: perldoc IO::Socket::INET

will get the socket library objects documentation. Or browse to cpan.org

Server: Choose a random port to listen on (higher than 1500) and start you server to listen for a connection (it should take a port number as the command line argument) note you will be waiting for links and then running webthumb, generating a jpg and then writing it back to the client. At that point you can bring down the connection. (for multiple links the client can just open a new connection.

Client: The client should also take port number, and host machine (where the server is running) as command line arguments and url and should write out the url, and then read back the binary data from the server. It should save it to a local file.

Example for the webthumb machine on a clic lab machine you would run:
the ampersand would launch it in the background (don’t do this until it really works) then log in to the solaris box (cluster.cs.columbia.edu) and test your program: ./client.pl 9000 vienna.clic.cs.columbia.edu
http://www.cnn.com
Summary: in theory you would end up with a www_cnn_com.jpg picture in the same directory when done.
Make sure you understand that when the server waits for a read from the client, it will wait there until it gets the read (including the endline), same for client….chomp is your friend there.

This can take a long time if you don’t understand the underlying concepts….be sure to ask plenty of questions…..

**Step4: Tie it all together**

Now tie the different pieces together and make sure it works over the web. In theory it should be able to generate thumbnails on the fly for new urls entered. To speed things up you should save snapshots in the local system so you don’t need to regenerate them on the fly.

To goal will be a system a user can log into and view, delete, and add urls. You will be graded 70% on the working system, and 20% for coding/code comments, and 10% for proper documentation.

If webthumb does not end up working properly you can still get full credit if you show your system would have worked correctly if webthumb behaved ☺ but then it should work with everything else.

Get or Post are your choice.