Scripting in Network Programming

Lecture 5(A)

Introdution to Scripting

- Scripting is the process of writing a script, which is a small, interpreted program that can carry out a series of tasks and make decisions based on specific conditions it finds.

- Being interpreted, when it is executed, it is carried out one line at a time, as opposed to a compiled program, which is the process of turning it into machine language before it is run.

Scripting in Network Programming

- Scripting can significantly ease the burden of network administration by automating certain tasks which are repetitive.

- For example, login scripts run every time a user logs in to the network and can perform tasks like mapping network drives for the user based on certain conditions, such as group membership.

- Another example might need to be carried out only once, such as a modification to the registry, but to a large number of servers that are widely distributed geographically. In a case like that, you could create and distribute a single script to run the task on each server.

Choices of Scripting Language

- Unix Shell Scripting
- Perl
- etc...

Introduction to Perl

- a high-level, general-purpose, interpreted, dynamic programming language

- borrow features from other programming languages including C, shell scripting (sh), AWK, and sed.

- provide powerful text processing facilities without the arbitrary data-length limits of many contemporary Unix command line tools, facilitating easy manipulation of text files

Perl Language

- http://learn.perl.org/
- http://www.tutorialspoint.com/perl/
- http://perl-begin.org/learn/
- http://www.tizag.com/perlT/

LWP Module

- Ref: http://search.cpan.org/dist/libwww-perl/lib/LWP.pm

- LWP - The World-Wide Web library for Perl
  - a set of Perl modules which provides a simple and consistent application programming interface (API) to the Internet

  - provide classes and functions that allow you to write network client scripts

  - contain modules that are of more general use and even classes that help you implement simple HTTP servers

HTTP-Style Communication

- LWP is based on HTTP-style communication for all protocols supported

    - Means all network scripts in LWP is based on the HTTP’s **request/response** paradigm, even if the script is written to access an FTP or mail server!

Request Object

- Class name: **HTTP::Request**

- The **HTTP::** prefix only implies that we use the HTTP model of communication. It does not limit the kind of services we can try to pass this *request* to.
  - We can send HTTP::Requests both to ftp and mail servers, etc.

- The main attributes of the request objects are:
  - **method** is a short string that tells what kind of request this is. The most common methods are GET, PUT, POST and HEAD.
  - **uri** is a string denoting the protocol, server and the name of the "document" we want to access. The uri might also encode various other parameters.
  - **headers** contains additional information about the request and can also used to describe the content. The headers are a set of keyword/value pairs.
  - **content** is an arbitrary amount of data.

Response Object

- Class name: **HTTP::Response**

- The main attributes of objects of this class are:
  - **code** is a numerical value that indicates the overall outcome of the request.
  - **message** is a short, human readable string that corresponds to the code.
  - **headers** contains additional information about the response and describe the content.
  - **content** is an arbitrary amount of data.

Response **code**

- Since we don't want to handle all possible code values directly in our programs, a LWP response object has methods that can be used to query what kind of response this is. The most commonly used response classification methods are:
  - **is_success**
    - The request was successfully received, understood or accepted.
  - **is_error**
    - The request failed. The server or the resource might not be available, access to the resource might be denied or other things might have failed for some reason.

User Agent

- Class name: **LWP::UserAgent**

- an interface layer between your application code and the network. Through this interface you are able to access the various servers on the network.

- The main method provided by this object is **request**(). This method takes an **HTTP::Request** object as argument and (eventually) returns a **HTTP::Response** object.

User Agent

- The user agent has many other attributes that let you configure how it will interact with the network and with your application.
  - **timeout** specifies how much time we give remote servers to respond before the library disconnects and creates an internal *timeout* response.
  - **agent** specifies the name that your application uses when it presents itself on the network.
  - **from** can be set to the e-mail address of the person responsible for running the application. If this is set, then the address will be sent to the servers with every request.
  - **parse_head** specifies whether we should initialize response headers from the <head> section of HTML documents.
  - **proxy** and **no_proxy** specify if and when to go through a proxy server. <URL:http://www.w3.org/History/1994/WWW/Proxies/>
  - **credentials** provides a way to set up user names and passwords needed to access certain services.

Script Demonstration