IESoc 2005 Course: Networking in UNIX
by Adrian Sai-wah TAM, IE99 (swtam9@ie.cuhk.edu.hk)
February 24, 2005

Outline:

1. TCP Theory
   - Stream of data flow
   - UDP = datagram, separate
   - Reliable (acknowledged)
   - Connection oriented (with sequence number, set-up procedure, shut-down procedure)
   - Three-way handshake: SYN - SYN/ACK - ACK - Start
     - We can distinguish the direction of flows
     - Firewall blocks SYN and this can tear down everything
   - Port number: 1-65535
     - See IANA for assignment
     - Common assignment: 80=web, 21=ftp, 20=ftp-data, 25=smtp, 443=SSL
   - Common port for proxy: 8080, 8081, 3128
   - One port for one connection (both in or out) at a time

2. UNIX Services
   - Services = "server"
   - A program to handle request/reply through network
   - Sit there and wait for clients to come in (listen)
   - Handle client requests and gives reply, then wait for another client
   - Never stops autonomously
   - Examples:
     - Web: Apache
     - Email: Postfix, Sendmail, Qmail
     - Database: Oracle, MySQL, PostgreSQL
     - Proxy: Squid
     - FTP: proftpd, wu-ftpd, vs-ftpd
   - Most of them are written in C/C++ in Unix environment
     - The programming API is called "BSD Socket", which is very standard and you must know it in depth!
     - In Windows, the API is exactly the same, with some extension which you normally won't use
     - In Unix, the similar API is used in Java, Perl, ...
   - Programming network server normally needs also
     - Threading
     - Infinite looping
     - Signal handling
       - e.g. Ctrl-C = Stop peacefully
       - e.g. Stop-able by some signaling mechanism, like kill command
     - String handling (less important)
     - I/O control (even less if your project is small)
     - IEG3310 project 1 already addressed these

3. Debugging Unix Service using Telnet command
   - Telnet Theory
     - TCP is a stream
     - Simple TCP stream:
       1. Server listens at a TCP port
       2. Client starts, bind a port
       3. Client connects to server
       4. Connection success
       5. Send data (!)
       6. Finish sending data and receving data
       7. Terminate connection
       8. Terminate completes, client quit
       9. Server continues to listen at the same port
     - All TCP program are the same
     - Only difference: (!)
       - TCP is a layer-4 protocol, it does not describe anything about (!)
       - The application protocol describes what happen at there
         - Syntax of the dialogue between client program and server program
     - Telnet: defined no syntax
       - Client side: Input = keyboard, output = screen
       - Server side: A program to digest the input and gives the output
       - example: **`telnet library.cuhk.edu.hk`**
   - HTTP protocol
     - Defined in RFC1945 (HTTP 1.0), RFC2616 (HTTP 1.1)
     - Request style: (1.0)

       ```
       GET <url> HTTP/1.0 <enter>
       <other request headers, e.g. I-M-S>
       <enter>
       ```

     - Request style: (1.1)

       ```
       GET <url> HTTP/1.1 <enter>
       SITE: <host name>
       <other request headers, e.g. I-M-S>
       <enter>
       ```

     - Reply style:

       ```
       HTTP/1.0 200 OK
       Date: Wed, 23 Feb 2005 11:06:26 GMT
       Content-Type: text/html
       <other reply headers>
       <enter>
       <data>
       ```

     - We can demo this by using Telnet
     - We (web clients) don't care what server do actually

- SMTP protocol
  - Similar
  - Example:

```
HELO ie.cuhk.edu.hk
MAIL FROM: swtam3@ie.cuhk.edu.hk
RCPT TO: swtam3@ie.cuhk.edu.hk
DATA
From: i_dont_exists@ie.cuhk.edu.hk
To: who_are_you@ie.cuhk.edu.hk
Subject: This is a test mail

Data is here
Another line of data
.
```

- HTTP Proxy
  - Same as HTTP
  - Request line:

```
GET <url> HTTP/1.0
```

  - In HTTP:

```
GET / HTTP/1.0
```

  - In Proxy:

```
GET http://www.yahoo.com/ HTTP/1.0
```

4. Socket Programming in Unix
   - Must read: W. Richard Stevens, Unix Network Programming, Prentice-Hall
   - Other:
     - http://www.ecst.csuchico.edu/~beej/guide/net/
       (Beej's Guide to Network Programming)
     - http://www.faqs.org/faqs/unix-faq/socket/
       (Unix-socket-faq for network programming)
     - http://www.lowtek.com/sockets/
       (Spencer's Socket Site)
   - C is more intuitive and more controllable
   - Other languages (e.g. Java, Perl) are more compact and write less lines

Reference URL:

1. http://www.davidreilly.com/java/java_network_programming/
   Code excerpt showing how to fetch a file from HTTP, see section 2.3
2. http://www.xbill.org/dnsjava/
   Complete web server products in Java, including proxy server

Code:

From ref.1, code to fetch a file from HTTP:

```
import java.net.*;
import java.io.*;

public class URLDemo {
    public static void main(String args[]) throws Exception {
        try {
            // Check to see that a command parameter was entered
            if (args.length != 1) {
                 // Print message, pause, then exit
                 System.err.println ("Invalid command parameters");
                 System.in.read();
                 System.exit(0);
            }

            // Create an URL instance
            URL url = new URL(args[0]);

            // Get an input stream for reading
            InputStream in = url.openStream();

            // Create a buffered input stream for efficency
            BufferedInputStream bufIn = new BufferedInputStream(in);

            // Repeat until end of file
            for (;;) {
                int data = bufIn.read();

                // Check for EOF
                if (data == -1)
                    break;
                else
                    System.out.print ( (char) data);
            }
        } catch (MalformedURLException mue) {
            System.err.println ("Invalid URL");
        } catch (IOException ioe) {
            System.err.println ("I/O Error - " + ioe);
        }
    }
}
```