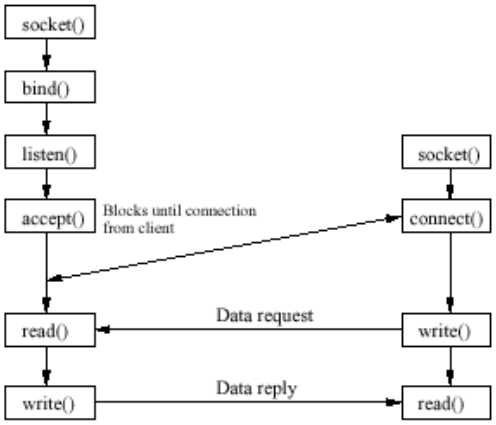
Using TCP/UDP in a Simple Application

Objective

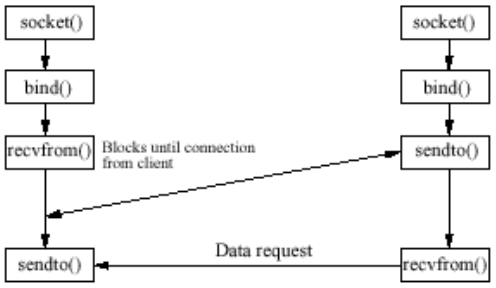
Introduction to the use of the TCP and UDP protocols in applications.

Background

You will take a look at the way that TCP and UDP connections are established. The typical order of events that occur in common TCP and UDP connections are shown in the following figure. These diagrams show the system calls made to establish connections when programming in C on a UNIX system. You will also examine a pair of Java programs that establish a connection via Sockets and exchange messages.



TCP Client/Server Communication Sequence



UDP Client/Server Communication Sequence

Preparation

Take a look at the source code of the Java programs that you will be executing (page 4).

Since the programs do not return the command prompt before exiting, you should run the client and the server in different windows. Remember that you can always terminate a program by pressing the Control-C key combination.

Procedure, part 1:

1. Login to the Sun workstation as `student`.
2. Open three terminal windows on the Sun workstation.
3. Run the MiniServer program in one window. (`java MiniServer portnumber`).
4. Run the MiniClient program in a 2nd window, connecting to the MiniServer on your machine (`java MiniClient ipaddress portnumber`).
5. Test the operation of the two programs and answer the questions below.

Questions

1. Does the MiniServer repeat the messages that you typed in the MiniClient window?

2. Does there seem to be a maximum message length that the Client / Server will accept?

3. What port number did you use for the Client / Server connection?

Procedure, part 2:

1. Close the MiniServer and MiniClient programs by hitting Control-C in each window
2. Use `su` to become `root` in one window and start a packet capture using:
   `/usr/local/sbin/tcpdump -w filename -s 500 ip`
   (where `filename` is the name of the file you are sending the packets to)
3. Restart the MiniServer program in one window using **segment# * 10000** as the port
4. Pick another Sun workstation and start a MiniClient that connects to their MiniServer.
   Note: only one MiniClient can connect to each MiniServer at a time.
5. Test the operation of the two programs by entering some text.
6. Close the MiniClient program by hitting Control-C in its window.
7. Stop the `tcpdump` capture by hitting Control-C in its window.
8. Answer the questions below (you may want to run ethereal to examine the capture file).

Questions

1. Do the Client and Server operate in the same way when you connect to another computer as they did when you ran them both on your computer?

2. How many packets were captured by tcpdump?

3. What application-level protocol was used for these connections (UDP or TCP)?

4. What port was used by your MiniClient when it connected to the other person's server?

```
//MiniServer.java - server that echos what it receives
import java.io.*;
import java.net.*;

class MiniServer{
  public static void main (String args[]) throws java.io.IOException
  {
    if (args.length != 1) { System.out.println("Usage: " +
                            "java MiniServer portnumber");
      System.exit(1);
    }
    int portnum = Integer.parseInt(args[0]);

    ServerSocket sock = null;
    try {
      sock = new ServerSocket(portnum);
    }
    catch (IOException e) {
      System.out.println("Could not listen on port: " + portnum + e);
      System.exit(1);
    }
    System.out.println("Now listening at port " + portnum);

    Socket clientSocket = null;
    try {
      clientSocket = sock.accept();
    }
    catch (IOException e) {
      System.out.println("Accept failed: " + portnum + ", " + e);
      System.exit(1);
    }
    BufferedReader input = new BufferedReader(
        new InputStreamReader(clientSocket.getInputStream()));

    PrintWriter output =
      new PrintWriter(clientSocket.getOutputStream());

    System.out.println("Connection established.");

    int i = 0;
    String line = input.readLine();         // waits for input
    while (line!=null) {
      System.out.println(line);
      i++;
      output.println("line " + i + ":" + line);
      output.flush();
      line = input.readLine();
    }
  }
}
```

```
//MiniClient.java - simple client for MiniServer
import java.io.*;
import java.net.*;

class MiniClient{
  public static void main (String args[])  throws java.io.IOException
  {
      if (args.length != 2) {
      System.out.println("Usage: " +
          "java MiniClient hostname portnumber");
      System.exit(0);
    }

    BufferedReader stdin = new BufferedReader(
       new InputStreamReader(System.in));

    int portnum = Integer.valueOf(args[1]).intValue();

    Socket sock = new Socket(args[0], portnum);

    BufferedReader input = new BufferedReader(
      new InputStreamReader(sock.getInputStream()));

    PrintWriter output =
      new PrintWriter(sock.getOutputStream());

    System.out.println("Connection established.");

    System.out.println("type some characters then" + " return:");

    String line = stdin.readLine();

    while (line != null) {
      output.println(line);
      output.flush();                 // forces it to write

      line = input.readLine();        // read from network

      System.out.println("got back:" + line);
      System.out.println("type some characters " + "then return:");
      line = stdin.readLine();
    }
  }
}
```

Web Server Installation on a Unix Workstation

Background

Most probably, you are familiar with the way software is installed on personal computers running the Windows operating system. The main difference between software designed for Windows and software designed for Unix is that usually Unix software comes in source code form - and you have to compile it for your specific machine and OS. The reason for this is that there are so many different architectures and flavors of Unix that preparation of precompiled binaries for all of them is not possible.

Usually every Unix OS comes with some kind of *package manager* which is a program used to install, uninstall, keep record and check integrity of other programs. Unfortunately, there is no standard in this area and all package managers are quite different from each other.

The following section describes how to install and uninstall the Apache Web Server on a SUN workstation.

Installing a Web Server on Solaris

Objective

In this exercise, you will learn how to install pre-compiled and packaged software for the Solaris Operating System. The particular program that you will install is the Apache web server.

Background

The Solaris Operating System comes with a propriety software manager developed by Sun Microsystems. The Solaris package manager consists of several programs (like `pkgadd`, `pkgrm`, `pkginfo`, ... etc.), which each perform some part of the package management function. For example, `pkgadd` is used to install software packages, `pkgrm` is used to uninstall the already installed ones and `pkginfo` gives information on what is currently installed on the system.

Preparation

Skim quickly through the man pages for the `pkgadd`, `pkgrm` and `pkginfo` commands as a preparation step. The apache file is in the `/student` directory.

Procedure

Go to the `/student` directory and confirm that you have a file called:

`apache-1.3.6-sol7-sparc-local`

1. Install the package with the following command (you must be `root` to do this):

   ```
   pkgadd -d ./apache-1.3.6-sol7-sparc-local
   ```

   You will be asked some questions during the installation procedure - answer them accordingly. Check whether the installation was successful by examining the output of the following command:

   ```
   pkginfo -l SMCapache
   ```

2. Using either `vi` or `pico`, open **`/usr/local/apache/conf/httpd.conf`** and change the following line:

   #ServerName hostname.domain.com

   to

   ServerName sun<*segment_number*>.netlab.cs.ucf.edu

   where <*segment_number*> is the LAN segment number for your workstation

3. Start the Apache web server with the following command:

   ```
   /usr/local/apache/bin/apachectl start
   ```

4. If the server was started successfully you should be able to connect to your Sun workstation with a web browser (either type `netscape` in a terminal window or right-click the mouse and click on `Links` and then `WebBrowser`). **Note:** the people at *www.sunfreeware.com* configured the default web access port to 8080 - you will have to specify it explicitly in the URL. Check that the web server is operational by trying to connect to the following address:

   `sun<segment_number>:8080`

   where <*segment_number*> is the same one you entered in step 2.

   If the Apache web page is displayed, your installation was successful.

5. Stop the Apache web server with the following command:

   ```
   /usr/local/apache/bin/apachectl stop
   ```

6. Uninstall the Apache web server with the following command:

```
pkgrm SMCapache
```

7. Check that the uninstall was successful by running the following command:

```
pkginfo -l SMCapache
```

   You should get a message like this:

```
ERROR: information for "SMCapache" was not found
```

Questions

1. Suppose that you want to determine whether the package "`SUNWgzip`" is installed. What command would you type to find out?

2. What is the version number of the "`SUNWgzip`" package (for the version installed on the Sun Workstation)?

3. What login account is allowed to install packages on a Solaris system?