Network Programming, Fall, 2003
Project 3: Batched Remote Access System (ras) in HTTP

9123571 Chung-Wei Hang 杭仲瑋
at Learning Lab, CIS, NCTU 交大資科學習科技實驗室
gis91571@cis.nctu.edu.tw

Nov, 24, 2003

Requirement 程式需求

In this homework, you are asked to write a batched ras HTTP system.

1

Write a CGI program to receive an HTTP request described as follows. The parameters of the HTTP request are:

h1=140.113.210.101	# the first ras server's IP.
p1=7070	# the first ras server's port.
f1=batch_file1	# the batch file name redirected to the first ras server.
h2=140.113.210.103	# the second ras server's IP.
p2=7070	# the second ras server's port.
f2=batch_file2	# the batch file name redirected to the second ras server.
h3=140.113.210.103	# the third ras server's IP.
p3=7070	# the third ras server's port.
f3=batch_file3	# the batch file name redirected to the third ras server.
h4=	# no more ras server.
h5=	# no more ras server.

Then, the CGI program connects to the three ras servers and then redirects the batch file (stored in the HTTP server) as input to these servers. When receiving messages, send these messages back to the browser as the returning web page. In each line, you must mark which server and line number. For example,

```
140.113.210.101:1> %                          # ls
140.113.210.102:1> bin/ test.html
140.113.210.102:2> %                          # ls bin
140.113.210.103:1> ls cat removetag number
140.113.210.101:2> %                          # printenv PATH
140.113.210.102:3> PATH=/bin
140.113.210.103:2> %                          # cat test.html
...
```

In order to see the effect of select, you can either use ¡br¿ to break each line or simply use the header ”Content-type: text/plain”.

Requirement

- In your web page, you must be able to accept at least 5 batches.
- You can use your own ras server (in HW#1) for testing.
- You must use nonblocking I/O (including connect()) for connections to ras servers.
- It is suggested that in your ras server you call ”sleep(10)” before accept(), in order to test non-blocking.
- It is suggested that you use a large batch file to test output blocking.
- You do not need to consider nonblocking connections back to servers.
- The CGI program must be working with Apache server in Unix.

2

Write a simple http server in Unix to support CGI.

Requirement

- This must work with TA’s CGI program and the previous CGI you wrote.

3

In Windows, write a console program for 1 working with httpsvr in Windows.

Requirement

- You can get httpsvr from MSDN. TA will also post.

4

In Windows SDK, write an asynchronous version of 1.

Requirement

- You should use WSAAsyncSelect or CAsyncSocket.
- TA will give you a sample code of using WSAAsyncSelect (next week) that you can use.

Due Date

- 11/24 for 1 2.
- 12/08 for 3 4.

Program Overview 程式概觀

Winsock CGI with select

這次作業目的是熟悉在 Windows 環境下利用 Visual Studio 開發 Winsock 的程式. 我使用的是 Visual C++ 6.0, 在 Project Setting 的設定上有下面幾點需要注意:

- Project 種類是 Win32 Console Application.
- Project Setting 的地方, Link 的 Object/Library Modules 的地方加上 "ws2_32.lib".

4.1 Source File: netprog3.cpp 原始碼

大部分的地方都和第一小題相同, 除了幾個地方不太一樣:

- **ws-util** 使用了 ”Winsock Programmer’s FAQ” 裡面的範例程式: ws-util.cpp 和 ws-util.h 兩個檔案, 裡面用到一個關於處理錯誤的函式.
- **WSAStartup** 在 Winsock 可以運作之前, 必須作一個啓動的動作: WSAStartup.
- **Exception FD_SET** 用來處理連線失敗的問題, 當 rasd server 不存在的時候, 會設定這個 fd_set 來讓我們處理.
- **WSAEWOULDBLOCK** 這是是在 non-blocking socket 的時候會發生的錯誤, 但是對 non-blocking I/O 來說這並不是錯誤, 只是表示目前 not available.
- **WSAGetLastError** 不同於 Unix 系統下面 errno 的設計, Winsock 採用的是 WSAGetLastError 來取得最後一個錯誤的代碼, 再由這個代碼來判斷錯誤的原因.
- 其他的地方大部分都是函式名稱的不同, 像是 ioctl/ioctlsocket, close/closesocket, . . . 等.

Winsock CGI with WSAAsyncSelect

不同於上一題, 這一題的希望我們利用 WSAAsyncSelect 來完成, 利用助教提供的 TAWinCGI 版本, 在 Visual C++ 6.0 下面新增一個 Win32 Application, Link 的地方同樣加上 ”ws2_32.lib”.

4.2 Source File: TAWinCGI.cpp 原始碼

原則上程式的演算法大致上一樣, 只是在 WSAAsyncSelect 模式下程式運作的流程不同. 主要有下面幾個重點:

- **WSAAsyncSelect** 這個函式對同一個 socket 第二次 WSAAsyncSelect 會覆蓋第一次的結果.
- **WSAGETSELECTERROR** 這函式提供我們在 WSAAsyncSelect 後產生的錯誤可以處理, 包括 rasd server 不存在.
- **Callback** WSAAsyncSelect 之後, 分別對於 FD_READ, FD_ACCEPT, FD_WRITE, FD_CLOSE, . . . 等不同的 event 去作處理, 利用 lParam 和 wParam 來對不同的錯誤和不同的連線作判斷.

- **FD_READ** 和之前幾小題都有些不同, 在 Unix Berkeley Socket 中, 當 select 設定了 read 的 fd_set 之後, 這表示 socket 有資料可以被讀, 但是這些資料必須要被讀完, 但是在 WSAAynsSelect 的機制下, FD_READ 如果資料沒有被 recv 完, 剩下的資料還是會觸發 FD_READ 一次.

Implementation Issues 實作要點

這兩小題所用到的方法大致和前兩題一樣.

Reference 參考資料

- Winsock Programmer's FAQ: "http://tangentsoft.net/wskfaq/"
- MSDN Library

Misc

- 在處理 read/recv 的時候, 可以利用 ioctl 加上 FIONREAD 來取得可以被讀的 byte 數, 方便處理.