Monday, 16 October 2017

EE 324, Programming Assignment #2

Concurrent client/server programs

1. Overview

In the previous homework, you developed the simple client/server programs that conduct a simple file transfer. But, in practice, servers should process multiple requests sent from many different clients (e.g., Web server). In this assignment, you need to design and implement concurrent client/server programs that enable multiple simultaneous requests by extending the previous work. There are *three* required programs; *(1) a process-based server, (2) an event-based server with I/O multiplexing, and (3) a client* that sends multiple requests to a server.

2. Process-based Server

- It automatically spawns a child process when a new TCP connection is requested.
- Therefore, you should use the "fork()" function to generate child process in your code.
- The spawned child process should follow *the scenario of the previous assignment* (i.e., handshake and file transfer with the defined protocol).
- But, the parent process should be still listening for other new connections after spawning the child process like typical Linux daemons.
- The binary must be named as "multi_server".
- **NOTE: You should reap all zombie processes by using "waitpid()"**

3. Event-based Server with I/O multiplexing

- It manages multiple socket descriptors (i.e., file descriptors) as a set and processes the requests whenever a descriptor state is ready.
- Thus, you should use the "select()" function to pick pending inputs.
- For the pending socket descriptor, your server should add a new socket descriptor (i.e., the return value of accept()) to the set.
- And, in the rest of the code, the server should follow *the scenario of the previous assignment* (i.e., handshake and file transfer with the defined protocol).
- Four macros FD_SET, FD_ISSET, FD_ZERO, and FD_CLR are also required to implement the I/O multiplexing (see the lecture 05 slide carefully).
- The binary must be named as "select_server"
- **NOTE: You should use "select()" function, not "epoll()"**

4. Client

- In this assignment, you need to slightly modify the previous client program.
- To generate simultaneous requests, we will use simple multiple threads using Posix Threads (Pthreads) Interface (see the lecture 06 slide carefully).
- And, the command line is changed as ("*./client [server_ip] [number_of_requests], e.g., ./client 127.0.0.1 100).*
- After execution, the main process generates threads as the number of request from the argument (put the sleep function for 100 milliseconds for each cycle).
- Then, each thread creates a new socket, connects the server.
- Next, using our defined protocol, it handshakes with the target server and sends a string (e.g., "I am a thread [thread_number]", and this is the different thing from the previous work).
- For example, after the handshake completed, the thread sends 0x0003 with the string "I am a thread [thread_number]", and 0x0004 with the file name " [thread_number].txt" to the server.
- The thread_number should be assigned from 1 to the number_of_request (e.g., assuming that number_of_request is 20, then the first thread_number is 1, and the next is 2,.., and so on).

5. Instruction for Submission

- You will be submitting one taball file to the KLMS website.
- Create a tarball (tar.gz) with all the source codes, README, and executables.
  - Tarball structure: (for program assignment 1)
    - Create a folder called "p2"
    - Put your source code in the folder named "p2/src"
    - Put your executable(s) in the folder named "p2/bin"
    - Put your make file "Makefile" in the root path - "p2/"
      - Make sure that your Makefile correctly complies your source code
    - readme.txt file goes to the root path - "p2/"
      - No README, No points; let us know how to run your program
    - install.sh goes to the root path as well - "p2/" (optional)
      - But, if your program needs a third party library, you should provide a script "install.sh"
    - Then, compress the entire "p2" folder into a single tarball; the name will be **"P2_yourIDnumber.tar.gz"**

- Write the concise comments in the source code to understand your program.

! **IMPORTANT 1: Please strictly follow the above structure and name policy; if not, TAs will not evaluate your program.**

! **IMPORTANT 2: Please make sure that there is no compile and execution error. TAs will not give you any score in case of the problems.**

6. Test Case

1) Process-based Server - 40 points

   I. Does the server spawn a child process using fork() whenever it receives a new connection? - 5 points

   II. Can the server process 20 (approximate) simultaneous connections from the client? - 35 points

2) Event-based Serve with I/O Multiplexing - 60 points

   III. Does the server use select() to pick pending connections? - 5 points

   IV. Does the server use bit sets using macros (i.e., FD_SET, FD_ISSET, FD_ZERO, and FD_CLR) to manage multiple socket descriptors? - 5 points

   V. Can the server process 200 (approximate) simultaneous connections from the client? - 50 points

7. Test Environment

- Language: C or C++
- Test O/S: **Ubuntu 14.04.5 LTS (Trusty Tahr) 64bit**
    - http://releases.ubuntu.com/14.04/
- If you need a VM, download using the following link.
    - http://nss.kaist.ac.kr/ee324.ova
    - username: ee324, password: ee324
    - Import the VM using VMware or VirtualBox
- **NOTE: We will not consider your compilation and execution problems due to the different OS versions.**

8. Due Date

- **11:59 PM, Oct. 31, 2017 (Friday)**

- The website automatically marks the time of the last submission/modification. (The website often becomes unavailable, so please make sure that you submit your assignment early)
- **10% late penalty per day**
- Hard deadline: **11:59 PM, Nov. 04. 2017 (Saturday)**
    - We will not receive additional submissions after the hard deadline.
    - Exceptions: documented medical/personal emergency
- Please DO NOT e-mail Prof. Shin or TAs to submit your assignments.

9. Plagiarism

- You can discuss with your colleagues, but you should turn in your own programs

    ✓ Copy and Paste

    - Will run plagiarism detection on source code
    - "Copy and paste" codes will get severely penalized
    - If detected, 0 point for all assignments (both providers and consumers)
    - But you will have a chance to defend yourself

10. Questions?

- Please email TAs to ask any questions.
    - Junsik Seo (js0780@kaist.ac.kr)
    - Jinwoo Kim (jinwoo.kim@kaist.ac.kr)