

CS420: Operating Systems

Lab 2: Threaded Server

Due: in class on Friday October 9th, 2009. For each 24 hours the solution is late 10 points will be deducted from the grade.

The Basic Assignment

Write a multi-threaded server which accepts network connections from clients that request a file to be read and returned to the client *via* a socket connection. Here are the basic requirements:

- Each client request to the server should spawn a new thread which in-turn services the request and then terminates.
- You can implement your solution in C or C++.
- The client code is provided in <http://cs.earlham.edu/~charliep/courses/cs420/code/client.c> and `client-server.h`. In addition to testing your server the client code also serves as a pattern for working with network sockets under Linux. The client's command line
- Your server's command line should look like so: `os-server <listen-port>`
- Instrument the server and client code so that all failure messages are written to `stderr`. Your server should trap the return values from *all* system calls and handle failures appropriately, including a message to `stderr`.
- Instrument the server and client code so that all success messages (client connect, thread creation, file transfer initiation and completion, client disconnect, thread termination) should be optionally written to `stderr`. This can be controlled with either a command line option or with a compile-time option.
- Since files are only read by the server no locking is required.
- Remember that user mode code can only open ports above a certain value under most Un*x implementations.

The Extras

For 10 points each complete either or both of the following:

1. Implement an internal command, say `/proc/os-server-info` that displays all active threads with their client IP numbers and requested file.
2. Implement the notion of threads living for more than one request, up to a certain adjustable maximum number of client requests serviced, before they are reaped. If you also implement `/proc/os-server-info` modify that code to also show the total number of requests serviced by each thread.

The Deliverables

To turn-in the assignment complete each of the following:

1. Write a short shell script that runs the client N times requesting the same file each time. Hint, redirecting the `stdout` of each process to `/dev/null` will make this work much better. The file should be at least 1MB of printable ASCII characters.
2. Start your server on an ACL machine. On *two* other ACL machines simultaneously start two runs of the client shell script with $N = 10$. If that works try 100 and 1000. Check the `stderr` and `stdout` (with logging enabled) of your client and server to make sure that all the requests were properly serviced.
3. Capture the shell output for a couple of reads of a short file using the client.
4. Capture shell output demonstrating any of the extra credit items you completed.
5. Print the source code to your server solution. Make sure any sections that handle extra credit items are marked as such. Turn-in the source and shell output printouts.

History

This exercise was originally developed by Dave Hovemeyer and later adapted by Charlie Peck. The sample client code came from the operating systems course at Benin School of Science and Engineering in Jerusalem.