Lasater, Doug Section: 504

CSCE 313: Machine Problem 5

# Files Being Turned In

* client.C
* semaphore.H & semaphore.C
* NetworkRequestChannel.H & NetworkRequestChannel.C
* dataserver.C
* makefile & .DS\_Store

# Testing Environment and Running the Code

* The code was developed in Notepad++
* The code was tested and run on the sun.cse.tamu.edu servers (Unix environment)
* To run the code you will need two putty windows open; one will act as the client, the other as the server.
* Once in the proper directory of the putty that will act as the server
  + make all
  + ./dataserver –p <port number> -b <backlog number>
    - Note: if simply sun with ./dataserver default will be p = 11335, blog = 200.
* Once in the proper directory of the putty that will act as the client,
  + ./client -n 10000 -b 10000 -w 65 –h localHost – p 11335
    - OR
  + ./client -n <any number> -b <any number> -w <any number> -h <IP address> -p <port number>

# Analysis of the Program

* The client starts off by processing the specific inputs (described above in the running of the code) for both the client and the data server.
* The client program sends n number of requests specified by the user to the bounded buffer for each person. These requests are removed from the buffer by the single event\_handler thread and sent to the data server.
* The information is sent to the data server with the use of NetworkRequestChannels which use network sockets to connect the client to the server. Each NetworkRequestChannel has its own socket file descriptor which the client writes to and then waits to read from once the server has replied back to the initial request.
* The replies read from the socket file descriptors are extracted and stored in the statistics buffer. The particular socket file descriptor that was successfully read from is then used again by simply writing another request to it and the process continues until all requests are processed.
* The replies are removed from the statistics buffer by three statistics threads and then are sorted and stored in their corresponding statistics array.
* The statistics for each person are then printed out along with the running time of the program.

# Results from Testing

* **System Performance**
  + The client program was tested with several different numbers of request channels and tested with varying sizes of the backlog buffer. The results are graphed and recorded below.
  + Increasing the number of backlogs does not have that great of effect on the server client system. However, increasing the number of NetworkRequestChannels results in faster results. After about 75 NetworkRequestChannels, the results flatten out and then after 125 NetworkRequestChannels the results begin to decrease.

# Table of Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Testing | Results | MP5 |  |
| Backlog B Size: | 200 | 400 | 600 | 10000 |
| # Request Chans: | Run Time | Run Time | Run Time | Run Time |
| 10 | 31.938277 | 32.185432 | 32.086624 | 32.265734 |
| 25 | 13.832465 | 13.806718 | 13.774056 | 13.350282 |
| 35 | 10.387217 | 10.398742 | 10.416578 | 10.670303 |
| 50 | 8.011232 | 7.971068 | 8.024433 | 8.851227 |
| 65 | 6.764382 | 6.659011 | 6.712947 | 7.895477 |
| 75 | 6.221874 | 6.201087 | 6.317396 | 7.705402 |
| 90 | 5.801572 | 5.886931 | 5.800618 | 7.174638 |
| 125 | 5.907081 | 6.635837 | 5.789633 | 6.259554 |

**Table 1.** Analysis of Results of MP5

# Graphs of Results