COMP 4320 - Programming Assignment 2

Demarco Guajardo: dag0047
Cale Johnson: bcj0022

The programs **do** work together.

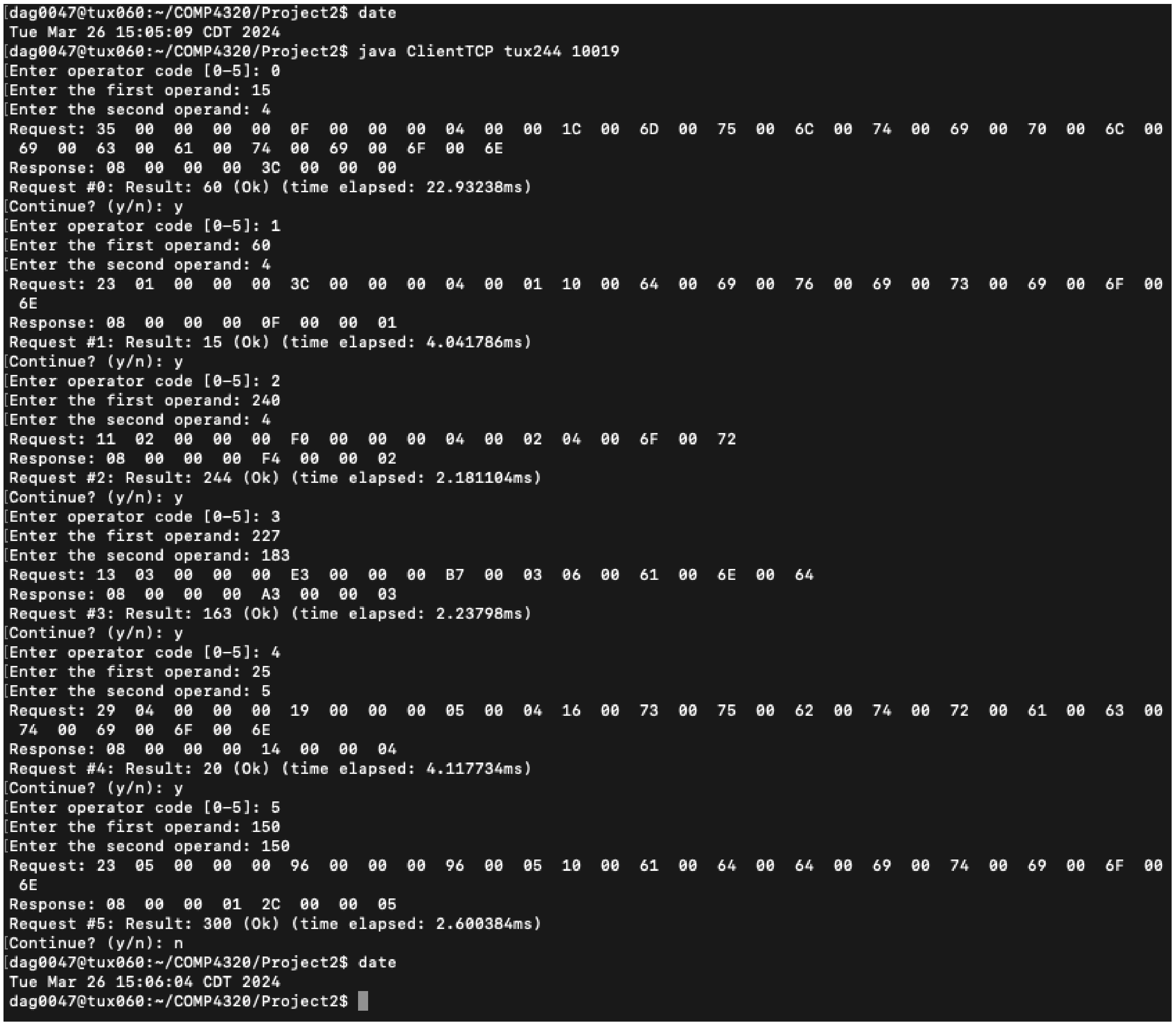
In order to run the programs, all 8 files must be uploaded to the same directory on a tux server. You can move them from your computer to a tux machine using sftp. All programs must then be compiled by entering javac <filename>. Then, on one command prompt (or terminal) logged into a tux server, enter “java ServerTCP 10019” and hit enter. Then on another command prompt (or terminal) on the same computer or a different one, enter “java ClientUDP tux<# of server tux machine> <path to other machine if on a different computer> 10019” Then follow the instructions it gives.

For example, in our situation, we both launched two tux computers (this is after moving all files to the tux and compiling them). One terminal was tux060 and the other was tux244.
On tux244, I ran the following command: “java ServerTCP 10019” and it awaited a request. On tux060, I ran the following command: “java ClientTCP tux244”. This led to a prompt to enter an operator code from 0 to 5. I selected 3. This led to a prompt asking to enter the first operand, to which I inputted 227. This led to a prompt asking to enter the second operand, to which I inputted 183. After that, the request I just inputted was printed in hexadecimal bytes. On the server terminal (tux244), the hexadecimal bytes of the request were also printed there, following an “user-friendly” display of the request, reading “Request #0: 227 & 183”. Back on the client terminal (tux060), the response was recorded as well as a “user-friendly” display of the response, reading: “Request #0: Result: 163 (Ok) (time elapsed: 22.451094ms)” and immediately after that was “Continue? (y/n): “. After selecting “y” it will allow you to repeat the process with a new operation code and two new operands. If you select “n”, the client side will immediately exit, and on the server side, you’ll get the following message before exiting: “User doen. Exiting.”

The next pages have screenshots of the client and server terminals, going through operations 0-5 before hitting “n” to quit requesting, and thus, exiting the program.

Note***: For the TML, we followed the output of the assignment instructions. For example, with “240 | 4”, the TML in the output was 11, but after counting each byte, it’s actually 17 (including TML itself). For “227 & 183”, the TML was 13, but there were 19 bytes. It appears the TML was removing 6 bytes, so I implemented that in the code since it wasn’t clear whether to follow the output or the actual number of bytes.

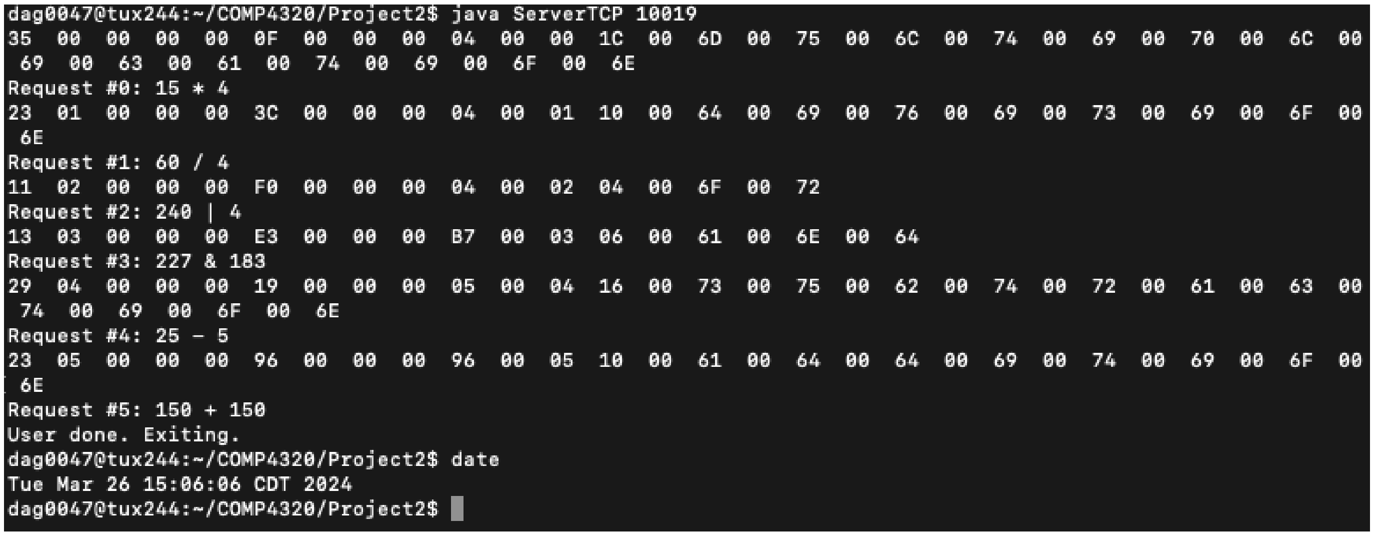
ClientTCP:



```
[dag0047@tux060:~/COMP4320/Project2$ date
 Tue Mar 26 15:05:09 CDT 2024
[dag0047@tux060:~/COMP4320/Project2$ java ClientTCP tux244 10019
[Enter operator code [0-5]: 0
[Enter the first operand: 15
[Enter the second operand: 4
 Request: 35  00  00  00  00  0F  00  00  00  04  00  00  1C  00  6D  00  75  00  6C  00  74  00  69  00  70  00  6C  00
  69  00  63  00  61  00  74  00  69  00  6F  00  6E
 Response: 08  00  00  00  3C  00  00  00
 Request #0: Result: 60 (Ok) (time elapsed: 22.93238ms)
[Continue? (y/n): y
[Enter operator code [0-5]: 1
[Enter the first operand: 60
[Enter the second operand: 4
 Request: 23  01  00  00  00  3C  00  00  00  04  00  01  10  00  64  00  69  00  76  00  69  00  73  00  69  00  6F  00
  6E
 Response: 08  00  00  00  0F  00  00  01
 Request #1: Result: 15 (Ok) (time elapsed: 4.041786ms)
[Continue? (y/n): y
[Enter operator code [0-5]: 2
[Enter the first operand: 240
[Enter the second operand: 4
 Request: 11  02  00  00  00  F0  00  00  00  04  00  02  04  00  6F  00  72
 Response: 08  00  00  00  F4  00  00  02
 Request #2: Result: 244 (Ok) (time elapsed: 2.181104ms)
[Continue? (y/n): y
[Enter operator code [0-5]: 3
[Enter the first operand: 227
[Enter the second operand: 183
 Request: 13  03  00  00  00  E3  00  00  00  B7  00  03  06  00  61  00  6E  00  64
 Response: 08  00  00  00  A3  00  00  03
 Request #3: Result: 163 (Ok) (time elapsed: 2.23798ms)
[Continue? (y/n): y
[Enter operator code [0-5]: 4
[Enter the first operand: 25
[Enter the second operand: 5
 Request: 29  04  00  00  00  19  00  00  00  05  00  04  16  00  73  00  75  00  62  00  74  00  72  00  61  00  63  00
  74  00  69  00  6F  00  6E
 Response: 08  00  00  00  14  00  00  04
 Request #4: Result: 20 (Ok) (time elapsed: 4.117734ms)
[Continue? (y/n): y
[Enter operator code [0-5]: 5
[Enter the first operand: 150
[Enter the second operand: 150
 Request: 23  05  00  00  00  96  00  00  00  96  00  05  10  00  61  00  64  00  64  00  69  00  74  00  69  00  6F  00
  6E
 Response: 08  00  00  01  2C  00  00  05
 Request #5: Result: 300 (Ok) (time elapsed: 2.600384ms)
[Continue? (y/n): n
[dag0047@tux060:~/COMP4320/Project2$ date
 Tue Mar 26 15:06:04 CDT 2024
 dag0047@tux060:~/COMP4320/Project2$ 
```

Below is the ServerTCP output.

<u>ServerTCP</u>:



```
dag0047@tux244:~/COMP4320/Project2$ java ServerTCP 10019
35  00  00  00  00  0F  00  00  00  04  00  00  1C  00  6D  00  75  00  6C  00  74  00  69  00  70  00  6C  00
 69  00  63  00  61  00  74  00  69  00  6F  00  6E
Request #0: 15 * 4
23  01  00  00  00  3C  00  00  00  04  00  01  10  00  64  00  69  00  76  00  69  00  73  00  69  00  6F  00
 6E
Request #1: 60 / 4
11  02  00  00  00  F0  00  00  00  04  00  02  04  00  6F  00  72
Request #2: 240 | 4
13  03  00  00  00  E3  00  00  00  B7  00  03  06  00  61  00  6E  00  64
Request #3: 227 & 183
29  04  00  00  00  19  00  00  00  05  00  04  16  00  73  00  75  00  62  00  74  00  72  00  61  00  63  00
 74  00  69  00  6F  00  6E
Request #4: 25 - 5
23  05  00  00  00  96  00  00  00  96  00  05  10  00  61  00  64  00  64  00  69  00  74  00  69  00  6F  00
 6E
Request #5: 150 + 150
User done. Exiting.
dag0047@tux244:~/COMP4320/Project2$ date
Tue Mar 26 15:06:06 CDT 2024
dag0047@tux244:~/COMP4320/Project2$ 
```

Data Analysis:

- The minimum round time was 2.18ms (this was request #2: “240 | 4”)
- The maximum round time was 22.93ms (this was request #0: “15 * 4”)
- The average round time was 6.35ms for 6 examples.