

## PROJECT 2

In order to run the project, “Makefile” includes the gcc commands as well as run scripts so that server and clients can be executed concurrently. When all clients finish their executions, “killall -9 server” command executes to terminate the server. This command becomes handy if you don't want to use ^D (Ctrl+D) command, which closes the terminal. This is exemplified in next page.

An example “Makefile” includes:

```
all: server client run
server: server.c
    gcc -Wall -o server server.c -lpthread -lrt -I.

client: client.c
    gcc -Wall -o client client.c -lrt -I.

run:
    ./server /ali &
    ./client /ali hello 1 file2 &
    ./client /ali hello 2 file1 file2
    killall -9 server
```

If you wish to run clients with different parameters you can change the Makefile or if you want to execute commands within the terminal by hand, you delete the **red** indicated lines. As:

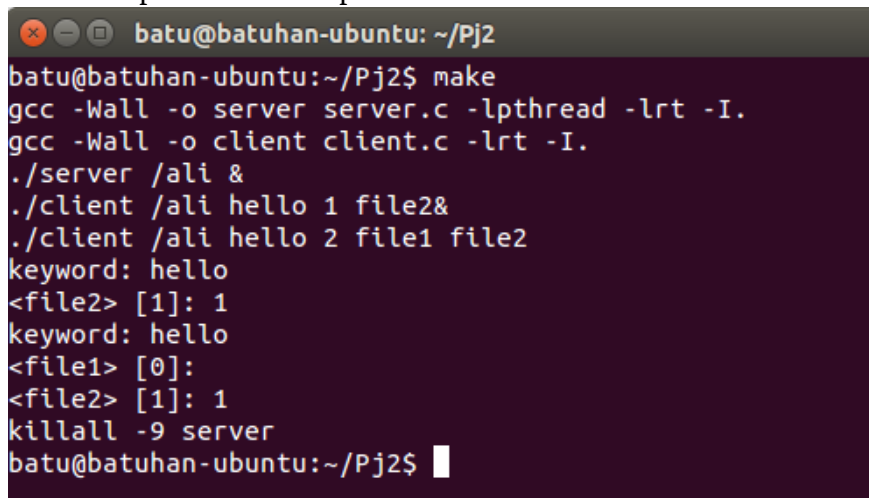
```
all: server client
server: server.c
    gcc -Wall -o server server.c -lpthread -lrt -I.

client: client.c
    gcc -Wall -o client client.c -lrt -I.
```

If you don't want server to continue running, you can simply invoke ^D (Ctrl+D) to close terminal and running server but if you don't want to close terminal you can use “killall -9 server” command to kill server.

### NOTES:

Since it was not mentioned in the homework documentation: I assumed the queue names are invoked with “/” character at the beginning. “/ali” is a queue name and “hello” is the keyword. A simple execution example with the output:



```
batu@batuhan-ubuntu: ~/Pj2
batu@batuhan-ubuntu:~/Pj2$ make
gcc -Wall -o server server.c -lpthread -lrt -I.
gcc -Wall -o client client.c -lrt -I.
./server /ali &
./client /ali hello 1 file2&
./client /ali hello 2 file1 file2
keyword: hello
<file2> [1]: 1
keyword: hello
<file1> [0]:
<file2> [1]: 1
killall -9 server
batu@batuhan-ubuntu:~/Pj2$
```

I prepared 10 different input files from file1 to file10 to test my program. Their content:

File1	File2	File3	File4	File5	File6	File7
	hello	asasd hello hello	hellbro asd hello hello	hello hello hi hello hi hello	hello bro hello mommy hello nice day isn't it hello everyone	what a wonderful world hello hi hello hello hello hi to you hello to me this makes no sense hellooooo

File8	File9	File10
hello asd nvja sd aib fash a ijsdn asdas dinasdi asdi masdiknasd nais osidnaosd nlasd aos jbdşnasjd anjsdab asdbasd ndasnd aslkdnalsj asjkldbaskdj hello hello hello asdjinsd hello asdlkin hello	hello hello hello nope nope nope hello nope hellooo helooppopjasdp hello posa minasd hello helloasdnlas nais osidnaosd nlasd aos jbdşnasjd anjsdab asdbasd ndasnd aslkdnalsj asjkldbaskdj hello hello hello asdjinsd hello asdlkin hello	hello hello hello nope nope nope hello nope hellooo helooppopjasdp hello posa minasd hello helloasdnlas nais osidnaosd nlasd aos jbdşnasjd anjsdab asdbasd ndasnd aslkdnalsj asjkldbaskdj hello hello hello asdjinsd hello asdlkin hello hello hello hello nope nope nope hello nope hellooo helooppopjasdp hello posa minasd hello anjsdab asdbasd ndasnd aslkdnalsj asjkldbaskdj hello hello hello asdjinsd hello asdlkin hello

I applied several experiments to see the varying execution times for clients from 1 client running to 5 clients running concurrently by using the above input files.

I used “time” command to see the execution time and my experiments are based on the real time value, which indicates the time between invocation of the make and its termination.

(Example experiment execution)

```

batu@batuhan-ubuntu: ~/Pj2
batu@batuhan-ubuntu:~/Pj2$ time make
./server /ali &
./client /ali hello 1 file2
keyword: hello
<file2> [1]: 1
killall -9 server

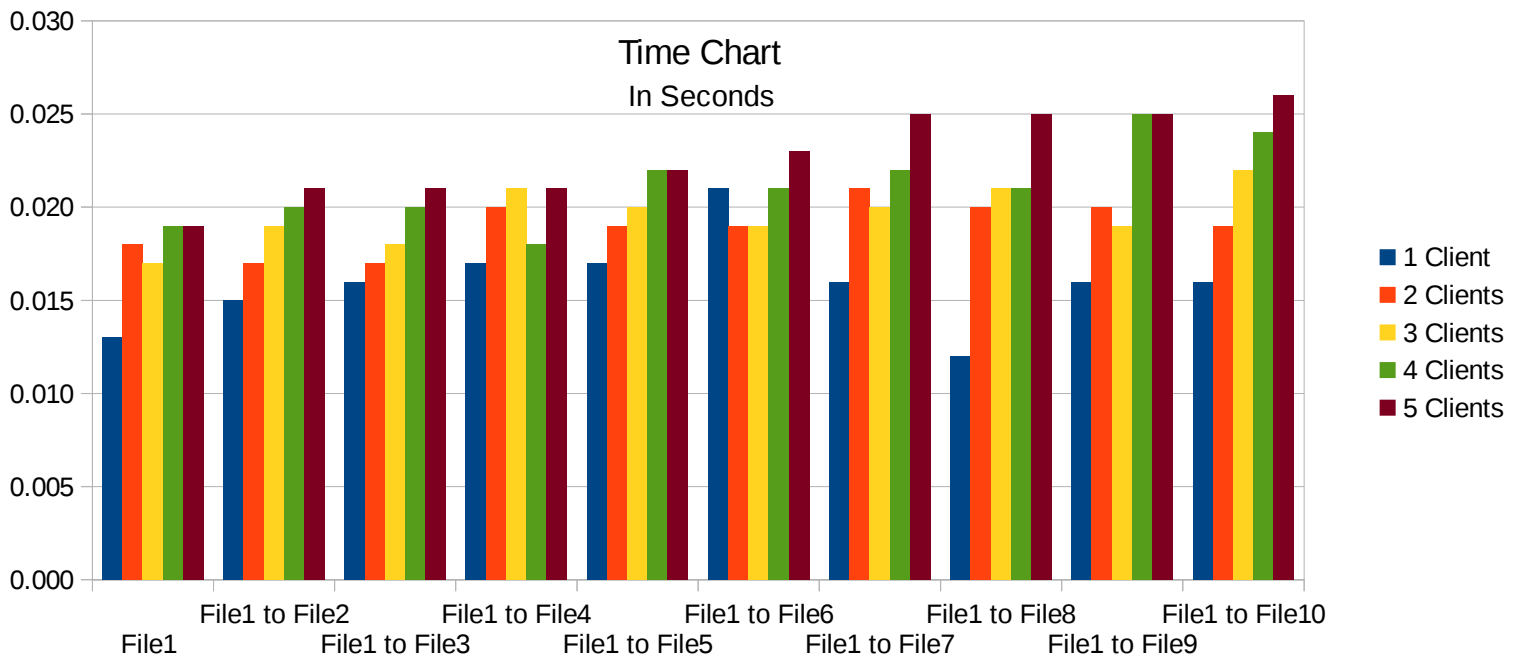
real    0m0.016s
user    0m0.005s
sys     0m0.011s
batu@batuhan-ubuntu:~/Pj2$

```

## Experiments

Experiment are done with 1 client running and 2,3,4 and 5 clients running concurrently with the given input files.

This chart shows the results of elapsed time for different number of clients given that passing different number of files to the server in seconds.



This is the data table of the above chart:

**Data Table**

	Categories	Y-Values	Y-Values	Y-Values	Y-Values	Y-Values
1	File1	0.013	0.018	0.017	0.019	0.019
2	File1 to File2	0.015	0.017	0.019	0.020	0.021
3	File1 to File3	0.016	0.017	0.018	0.020	0.021
4	File1 to File4	0.017	0.020	0.021	0.018	0.021
5	File1 to File5	0.017	0.019	0.020	0.022	0.022
6	File1 to File6	0.021	0.019	0.019	0.021	0.023
7	File1 to File7	0.016	0.021	0.020	0.022	0.025
8	File1 to File8	0.012	0.020	0.021	0.021	0.025
9	File1 to File9	0.016	0.020	0.019	0.025	0.025
10	File1 to File10	0.016	0.019	0.022	0.024	0.026

According to this Time Chart and its Data Table; as the number of files increases it takes more time to execute as expected but when there are multiple clients running concurrently, execution time of all does not vary too much. This is because the program is thread-safe and all threads are running concurrently and parallel.

A sample output for the 2 clients running concurrently and parallel, passing 10 files as argument to client:

```
batu@batuhan-ubuntu: ~/Pj2
batu@batuhan-ubuntu:~/Pj2$ time make
./server /ali &
./client /ali hello 10 file1 file2 file3 file4 file5 file6 file7 file8 file9 file10 &
./client /ali hello 10 file1 file2 file3 file4 file5 file6 file7 file8 file9 file10
keyword: hello
<file1> [0]:
<file2> [1]: 1
<file3> [3]: 1 2 3
<file4> [4]: 1 2 3 3
<file5> [4]: 1 2 3 4
<file6> [2]: 1 3
<file7> [3]: 1 2 2
<file8> [6]: 1 11 11 11 12 12
<file9> [22]: 1 1 1 3 5 6 12 12 12 13 13 14 14 14 16 18 19 25 25 25 26 26
<file10> [22]: 1 1 1 3 5 6 12 12 12 13 13 14 14 14 16 18 19 22 22 22 23 23

keyword: hello
<file1> [0]:
<file2> [1]: 1
<file3> [3]: 1 2 3
<file4> [3]: 1 3 3
<file5> [4]: 1 2 3 4
<file6> [2]: 1 3
<file7> [5]: 1 2 2 2 3
<file8> [6]: 1 11 11 11 12 12
<file9> [20]: 1 1 1 3 5 6 12 12 12 13 13 14 14 14 16 18 19 25 25 26
<file10> [21]: 1 1 3 5 6 12 12 12 13 13 14 14 14 16 18 19 22 22 22 23 23

killall -9 server

real    0m0.021s
user    0m0.000s
sys     0m0.015s
batu@batuhan-ubuntu:~/Pj2$
```

## Memory Leak

I also checked clients for any memory leaks by using “valgrind”

A sample client invoked with memory leak check:

```
batu@batuhan-ubuntu: ~/Pj2
batu@batuhan-ubuntu:~/Pj2$ make
./server /ali &
batu@batuhan-ubuntu:~/Pj2$ valgrind ./client /ali hello 1 file5
==8401== Memcheck, a memory error detector
==8401== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==8401== Using Valgrind-3.10.0.SVN and LibVEX; rerun with -h for copyright info
==8401== Command: ./client /ali hello 1 file5
==8401==
==8401== Conditional jump or move depends on uninitialised value(s)
==8401==    at 0x508AA03: vfprintf (vfprintf.c:1661)
==8401==    by 0x5093498: printf (printf.c:33)
==8401==    by 0x400E12: main (in /home/batu/Pj2/client)
==8401==
keyword: hello
<file5> [4]: 1 2 3 4

==8401==
==8401== HEAP SUMMARY:
==8401==    in use at exit: 0 bytes in 0 blocks
==8401==    total heap usage: 5 allocs, 5 frees, 16,218 bytes allocated
==8401==
==8401== All heap blocks were freed -- no leaks are possible
==8401==
==8401== For counts of detected and suppressed errors, rerun with: -v
==8401== Use --track-origins=yes to see where uninitialised values come from
==8401== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
batu@batuhan-ubuntu:~/Pj2$
```

As indicated above output, all clients terminate without any memory leak.