CMPE207-HOMEWORK2
REPORT

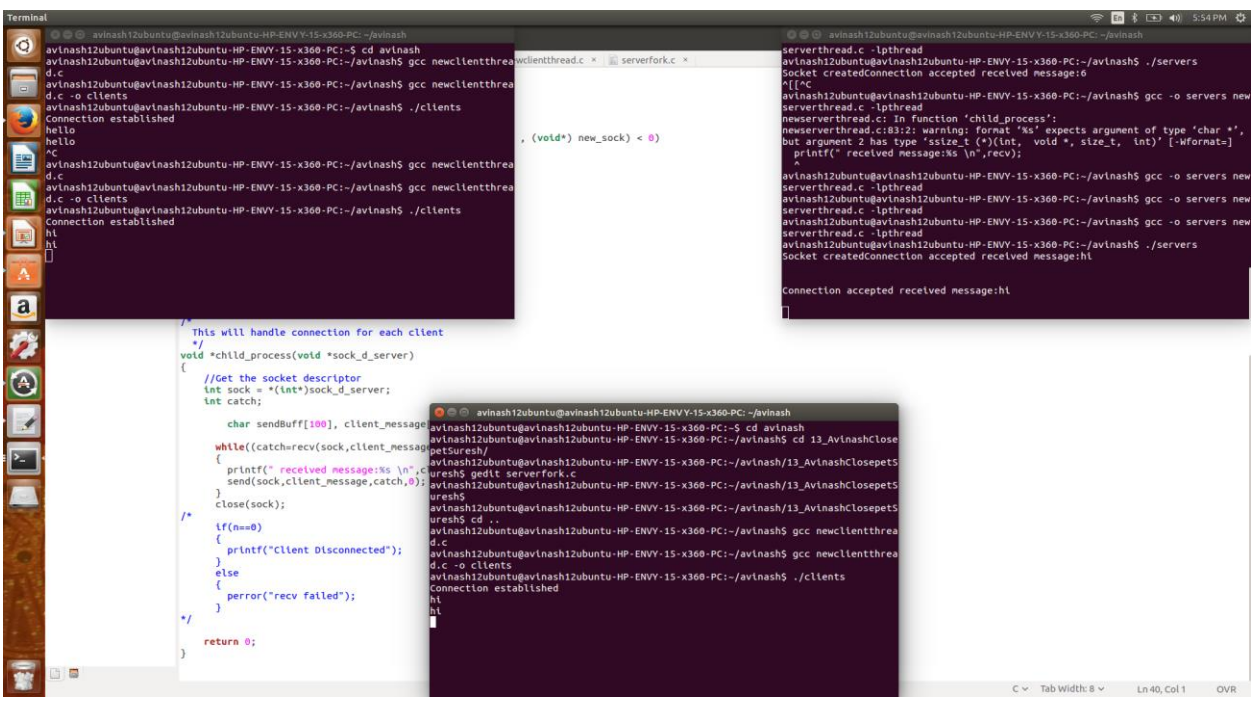
Q1). Using fork () for concurrent server and client model. ?
ANS:

```
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC: ~/avinash/Avinash_13_HW2
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ cd Avinash_13_HW2/
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/Avinash_13_HW2$ ls
13_HW2_Q1forkclient      13_HW2_Q1selectserver.c  13_HW2_Q2daytime.o
13_HW2_Q1forkclient.c    13_HW2_Q1selectserver.o  13_HW2_Q3echoforkserver
13_HW2_Q1forkclient.o    13_HW2_Q1threadclient    13_HW2_Q3echoforkserver.c
13_HW2_Q1forkserver      13_HW2_Q1threadclient.c  13_HW2_Q3echoforkserver.o
13_HW2_Q1forkserver.c    13_HW2_Q1threadclient.o  13_HW2_Q3echothreadserver
13_HW2_Q1forkserver.o    13_HW2_Q1threadserver    13_HW2_Q3echothreadserver.c
13_HW2_Q1selectclient    13_HW2_Q1threadserver.c  13_HW2_Q3echothreadserver.o
13_HW2_Q1selectclient.c  13_HW2_Q1threadserver.o  a.out
13_HW2_Q1selectclient.o  13_HW2_Q2daytime         Makefile
13_HW2_Q1selectserver    13_HW2_Q2daytime.c       Makefile~
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/Avinash_13_HW2$ gcc
13_HW2_Q1forkserver.c
gcc13_HW2_Q1forkserver.c: command not found
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/Avinash_13_HW2$ gcc
 13_HW2_Q1forkserver.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/Avinash_13_HW2$ gcc
 13_HW2_Q1forkserver.c -o king
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/Avinash_13_HW2$ ./k
ing
succes/nentered/ncatch=50 the message is : hi server
```

```
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC: ~/avinash
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~$ cd avinash
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc 13_HW2_Q1forkc
lient.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc 13_HW2_Q1forkc
lient.c -o tuma
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./tuma
 the sending message is: hi server

Message=hi server
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc 13_HW2_Q1forkc
lient.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc 13_HW2_Q1forkc
lient.c -o queen
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./queen
 the sending message is: hi server

Message=i got your messasge
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$
```

The above screen shot shows the output for concurrent server and client using the fork() system call.

Q1.4). Using thread () for concurrent server and client?
ANS:



The above screen shot shows the server and client communication using pthread_create which creates multiple thread to handle multiple clients.

Q1.2). Using select system call to implement server and client?

ANS:

```
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC: ~/avinash
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc 13_HW2_Q1selec
tclient.c -o point
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./ point
bash: ./: Is a directory
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./point
enter the message you want to send
send: hello
 server msg:  thank you
thank you

enter the message you want to send
send:
```

```
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC: ~/avinash
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc  13_HW2_Q1sele
ctserver.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc  13_HW2_Q1sele
ctserver.c -o shiva
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./shiva
server is ready
```

The above screen shot shows the output of concurrent server implemented using select system call.

Q1.3) Using synchronous IO except select we need to implement server and client?

ANS:

```
vinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/Avinash_13_HW2$ gcc
signalserver.c
vinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/Avinash_13_HW2$ gcc
signalserver.c -o giri
vinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/Avinash_13_HW2$ ./g
ri

ew client
ata Received from Client to Server is :
grgrttrt
```

```
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/Avinash_13_HW2$ gcc
 signalclient.c -o avi
signalclient.c: In function ‘main’:
signalclient.c:29:12: warning: assignment makes integer from pointer without a c
ast
 buffer[512]="NULL";
            ^
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/Avinash_13_HW2$ ./a
vi
grgrgttrt
hello
```

The above output shows the server and client communication using fcntl system call an synchronous IO. The clients sends a message which is reflected in the server.

Q2). Modify TCPdaytimed.c in textbook so that it creates a new child process to handle each incoming request. The thread converts the time string into the similar format as above and returns the revised string. Explain your code.

ANS:

```
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC: ~/avinash
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/13_AvinashClosepetS
uresh$ cd avinash
bash: cd: avinash: No such file or directory
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash/13_AvinashClosepetS
uresh$ cd ..
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gedit daytimefork.
c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gedit clientfork.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc daytimefork.c
daytimefork.c: In function ‘TCPdaytimed’:
daytimefork.c:26:1: warning: ‘return’ with a value, in function returning void
 return 0;
 ^
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc daytimefork.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc daytimefork.c
-o shama
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./shama
succes/n
```

```
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC: ~/avinash
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~$ cd avinash
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc clientfork.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc client
client.c          clientfork.c~      clientselectnew.c  clientudp.c
client.c~         clienthread.c      clientthread.c     clientudp.c~
clientfork.c      clientselect.c~    clientthread.c~
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc clientfork.c -
o sita
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./sita
 connect/nMessage=i got your messasge
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc clientfork.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc clientfork.c -
o baby
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./baby
 connect/nMessage=Tue Mar 29 10:21:54 2016
j{avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$
```

The above screen shot clearly shows the output of the TCPdaytimed.c with the help of concurrent fork program.

The output is clearly as seen: Tue Mar 10:21:54 2016.The TCPdaytimed.c function returns the time which is not read able by the humans, so in order to convert it we use ctime. Then the time is getting printed upon request.

Q3). Compare the multi-threaded, concurrent server in Chapter 12 to the implement in Chapter 11 that uses multiple, singly-threaded processes. Which executes faster? How does the running time vary with the number of concurrent connections? You need to draw a graph or table to show different execution time for each server?

ANS:

FORK(multi process echo server and client)

```
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC: ~/avinash
13_AvinashClosepetSuresh.zip  13_HW2_Q1threadclient.c
13_HW2_Q1forkclient.c         13_HW2_Q1threadserver.c
13_HW2_Q1forkserver.c         13_HW2_Q2daytime.c
13_HW2_Q1selectclient.c       13_HW2_Q2daytime.c~
13_HW2_Q1selectclient.c~      13_HW2_Q3echoforkserver.c
13_HW2_Q1selectserver.c       13_HW2_Q3echothreadserver.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc 13_HW2_Q3echof
orkserver.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc 13_HW2_Q3echof
orkserver.c -o gani
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./gani
succes/nentered/ncatch=50 the message is : hi server

 the time taken to handle clients is :0.000022
succes/nentered/ncatch=50 the message is : hi server

 the time taken to handle clients is :0.000020
succes/nentered/ncatch=50 the message is : hi server

 the time taken to handle clients is :0.000039
succes/nentered/ncatch=50 the message is : hi server

 the time taken to handle clients is :0.000047
```

```
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC: ~/avinash
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~$ cd avinash
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc 13_HW2_Q1forkc
lient.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc 13_HW2_Q1forkc
lient.c -o subba
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./subba
 the sending message is: hi server

Message=hi server
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$
```

The above screen shot gives the time for an echo server to Handel upto 4 clients and the time to handle those clients are shown above. This is a multi-process echo server, and its timings.

THREAD (threading echo server and client)

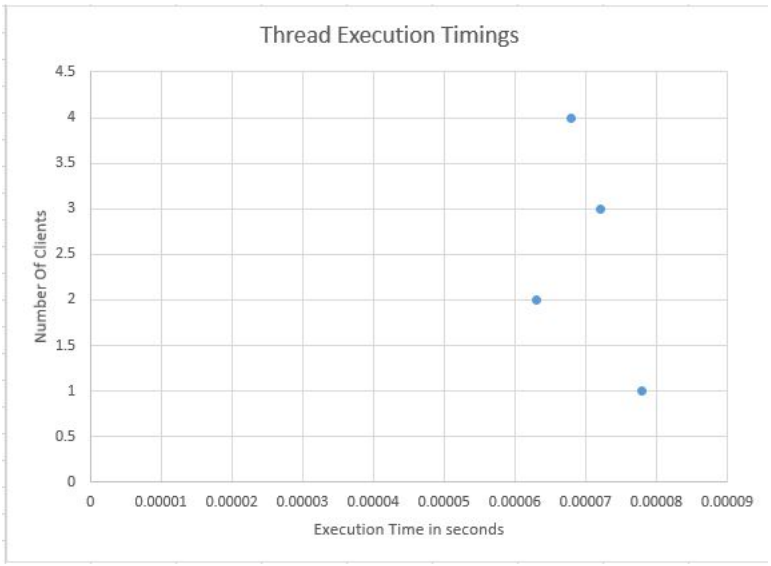
The screen shot provided below shows the server handling multiple clients using threads. Here it is clearly shown the timing for each of the clients the server handles.

The difference between the two is explained by using a graph as plotted below.

```
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC: ~/avinash
13_HW2_Q3echothreadserver.c:(.text+0xea): undefined reference to `pthread_create
'
collect2: error: ld returned 1 exit status
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc -pthread  13_H
W2_Q3echothreadserver.c
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ gcc -pthread  13_H
W2_Q3echothreadserver.c -o sujatha
avinash12ubuntu@avinash12ubuntu-HP-ENVY-15-x360-PC:~/avinash$ ./sujatha
Connection accepted received message:hello

 the time taken to handle clients is :0.000078
Connection accepted received message:hello

 the time taken to handle clients is :0.000063
Connection accepted received message:where

 the time taken to handle clients is :0.000072
 received message:hello

 the time taken to handle clients is :0.001347
Connection accepted received message:hello

 the time taken to handle clients is :0.000068
```

The below graph shows the different execution time of both echo servers using fork (multi-process) and Thread (multi thread).

Execution time of server using fork



Execution time of server using Thread

According to the graph as we can easily find the difference between each execution time. We can clearly show that as the number of clients get added the process that are getting created to handel that in the fork takes more time . As process gets created in more time when compared to threads.

As in 2nd graph we can see that the threads that are created to Handel the clients take better times when compared to process. As a result threads are faster and better that process.