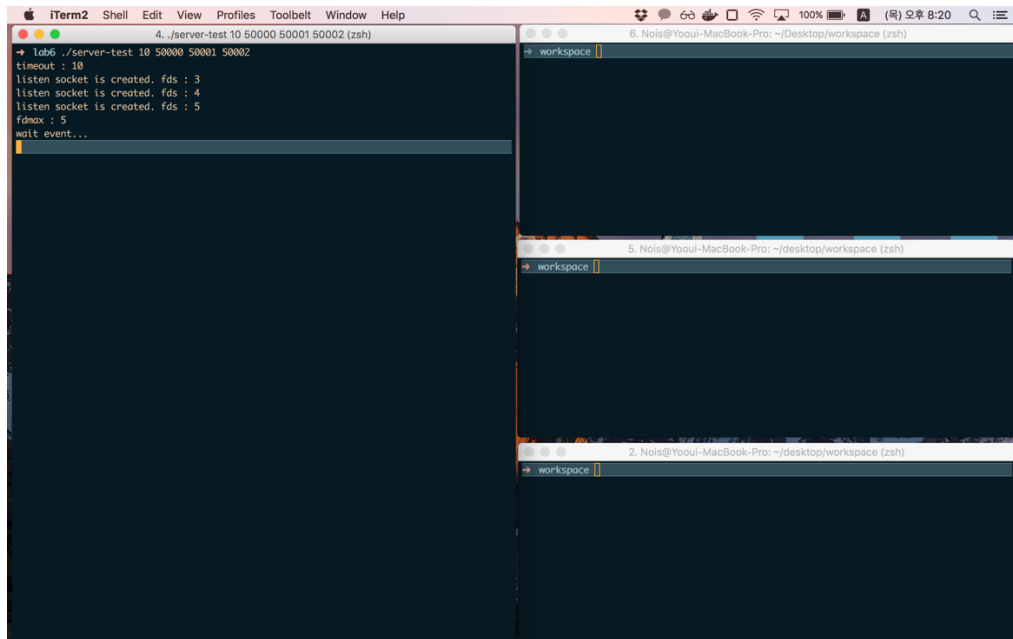


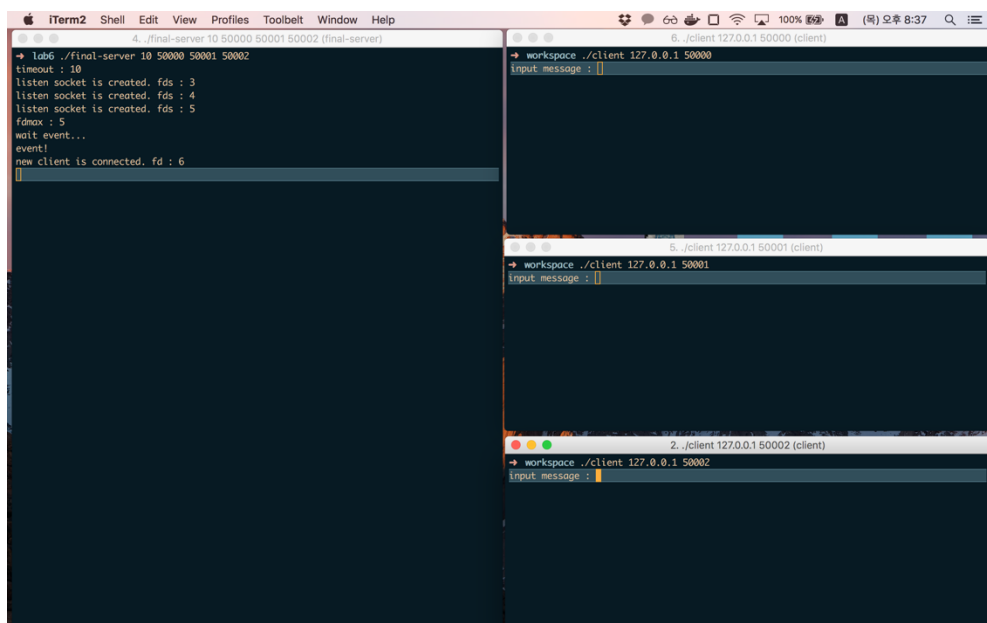
맥북(OS X) 에서 로컬에서 클라이언트와 서버를 실행시켜 얻은 결과임을 밝힌다.

1. TCP-Echo-Server

1번 TCP Echo Server의 경우, listen socket을 여러 개 만들고, 각각의 클라이언트와 연결시키는 방법을 사용한다. 먼저 그림을 보자.

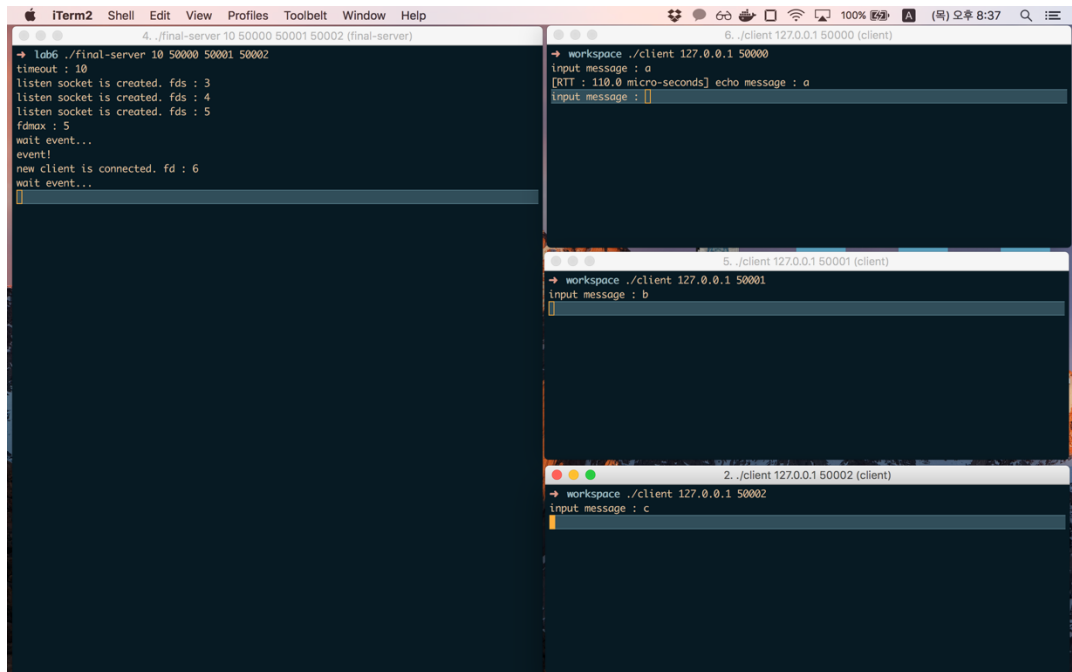


필자는 3개의 listen 소켓을 생성하여 각각 50000, 50001, 50002의 포트번호를 부여하였다. 또한 select의 timer를 10초로 설정하였다. 실제로 왼쪽의 서버를 보면, listen 소켓의 file descriptor가 각각 3, 4, 5로 새롭게 생성되었음을 확인할 수 있다.



다음으로, 오른쪽의 세 클라이언트에서 서버에 connect 요청을 하였다. 위의 그림에서 확인할

수 있듯이, 가장 위의 클라이언트. 즉, 가장 먼저 connect요청을 한 클라이언트만 정상적으로 접속됨을 확인할 수 있었다.



```
4. ./final-server 10 50000 50001 50002 (final-server)
→ lab6 ./final-server 10 50000 50001 50002
timeout : 10
listen socket is created. fds : 3
listen socket is created. fds : 4
listen socket is created. fds : 5
fdmax : 5
wait event...
event!
new client is connected. fd : 6
wait event...

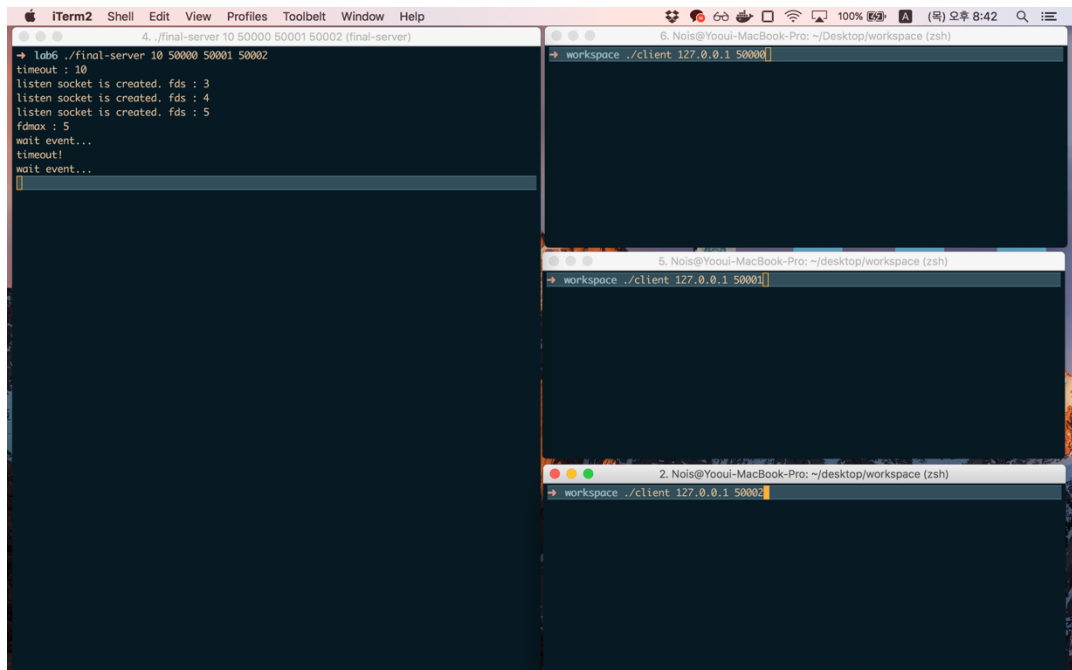
6. ./client 127.0.0.1 50000 (client)
→ workspace ./client 127.0.0.1 50000
input message : a
[RTT : 110.0 micro-seconds] echo message : a
input message :

5. ./client 127.0.0.1 50001 (client)
→ workspace ./client 127.0.0.1 50001
input message : b

2. ./client 127.0.0.1 50002 (client)
→ workspace ./client 127.0.0.1 50002
input message : c
```

메시지의 송수신 역시 첫번째 클라이언트에서만 정상적으로 이루어짐을 확인할 수 있다.

그리고 10초동안 어떠한 이벤트도 발생하지 않으면, time out 이벤트가 발생하여 time out! 이라는 메시지를 출력함을 확인할 수 있다.



```
4. ./final-server 10 50000 50001 50002 (final-server)
→ lab6 ./final-server 10 50000 50001 50002
timeout : 10
listen socket is created. fds : 3
listen socket is created. fds : 4
listen socket is created. fds : 5
fdmax : 5
wait event...
timeout!
wait event...

6. Nois@Yooul-MacBook-Pro: ~/Desktop/workspace (zsh)
→ workspace ./client 127.0.0.1 50000

5. Nois@Yooul-MacBook-Pro: ~/Desktop/workspace (zsh)
→ workspace ./client 127.0.0.1 50001

2. Nois@Yooul-MacBook-Pro: ~/Desktop/workspace (zsh)
→ workspace ./client 127.0.0.1 50002
```

The screenshot shows three terminal windows. The left window, titled '4. ./final-server 10 50000 50001 50002 (final-server)', displays the server's execution: it sets a 10-second timeout, creates three listen sockets (fds 3, 4, 5), and waits for events. A new client connects on fd 6, and the server enters a loop of waiting for events and processing them. The right side shows two client windows. The top client window, titled '6. Nois@Yooil-MacBook-Pro: ~/Desktop/workspace (zsh)', shows the command 'workspace ./client 127.0.0.1 50000' and a series of echo messages: 'a', 'aa', 'ab', 'ab', 'czvz', 'czvz', and '^C'. The bottom client window, titled '5. ./client 127.0.0.1 50001 (client)', shows the command 'workspace ./client 127.0.0.1 50001' and an empty input message field. A third client window is partially visible at the bottom, titled '2. ./client 127.0.0.1 50002 (client)', showing the command 'workspace ./client 127.0.0.1 50002' and an empty input message field.

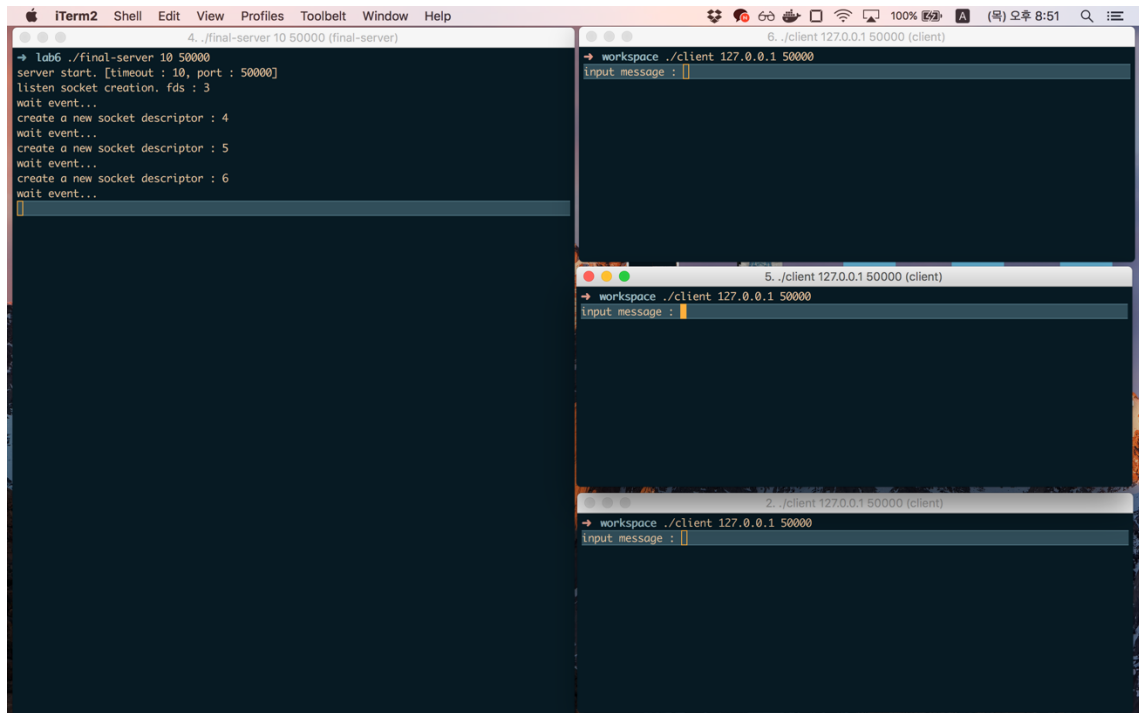
그리고, 첫 번째 클라이언트에서 `ctrl + c`를 통해 종료하면, 해당 클라이언트를 `close`함을 확인할 수 있었다. (fd가 6으로 일치한다.)

2. TCP Echo Server2

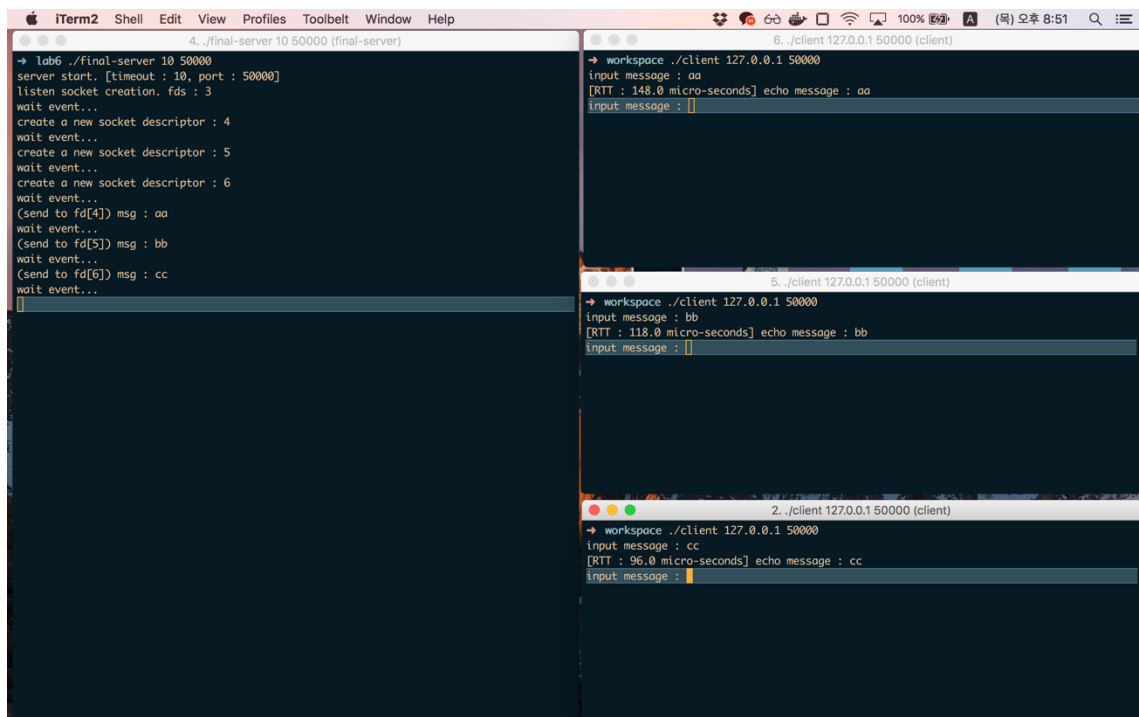
TCP Echo Server2는, listen socket을 하나만 두어 클라이언트의 `connect`요청을 받으면 새로운 디스크립터를 별도로 생성하여 해당 소켓과 연결하여 말 그대로 listen 소켓은 `connect` 요청만 수행하도록 하는 방법이다. listen 소켓은 'SO_REUSEADDR' 옵션을 활성화하였기 때문에 하나의 포트만 사용하여도 문제가 없다. 아래 그림을 보자.

The screenshot shows three terminal windows. The left window, titled '4. ./final-server 10 50000 (final-server)', displays the server's execution: it sets a 10-second timeout, port 50000, and creates one listen socket (fd 3). It then enters a loop of waiting for events. The right side shows three client windows, all titled '6. Nois@Yooil-MacBook-Pro: ~/Desktop/workspace (zsh)'. Each window shows the command 'workspace' followed by a prompt. The windows are stacked vertically, with the top one showing 'workspace' and the others showing 'workspace' followed by a prompt.

timeout 이벤트를 위해 10초, 그리고 50000번 포트를 열었다. 때문에 listen socket은 하나만 생성되었음을 확인할 수 있다.



그리고 세 클라이언트에서, 접속을 시도하면 새로운 소켓을 생성하고 각각의 클라이언트와 연결한다.



또한 모든 클라이언트와 성공적으로 메시지를 주고받음을 확인할 수 있다.

The screenshot shows four terminal windows. The leftmost window, titled '4. ./final-server 10 50000 (final-server)', shows the server's execution: it starts, listens on port 50000, creates three descriptors (4, 5, 6), and sends messages 'aa', 'bb', and 'cc' to them respectively. The other three windows on the right are client windows. The top one, '6. Nois@Yooul-MacBook-Pro: ~/Desktop/workspace (zsh)', shows a client sending 'aa' and receiving an echo. The middle one, '5. Nois@Yooul-MacBook-Pro: ~/desktop/workspace (zsh)', shows a client sending 'bb' and receiving an echo. The bottom one, '2. Nois@Yooul-MacBook-Pro: ~/desktop/workspace (zsh)', shows a client sending 'cc' and receiving an echo. All clients are using the command 'workspace ./client 127.0.0.1 50000'.

그리고 세 클라이언트가 종료하면 해당 소켓을 닫음을 확인할 수 있다.

This screenshot shows the same four terminal windows after the clients have finished. The server window on the left now shows the closing of descriptors 4, 5, and 6, followed by the creation of new descriptors with the same numbers. The three client windows on the right now show the prompt 'input message : ' without any further output, indicating they have finished their execution and closed their sockets.

그리고 다시 클라이언트에서 접속을 시도하면 소켓 디스크립터를 7, 8, 9로 사용하지 않고 전에 사용하던 4, 5, 6번의 디스크립터를 재사용함을 확인할 수 있다.

```
→ lab6 ./final-server 10 50000
server start. [timeout : 10, port : 50000]
listen socket creation. fds : 3
wait event...
create a new socket descriptor : 4
wait event...
create a new socket descriptor : 5
wait event...
create a new socket descriptor : 6
wait event...
(send to fd[4]) msg : aa
wait event...
(send to fd[5]) msg : bb
wait event...
(send to fd[6]) msg : cc
wait event...
close fd : 4
wait event...
close fd : 5
wait event...
close fd : 6
wait event...
create a new socket descriptor : 4
wait event...
create a new socket descriptor : 5
wait event...
create a new socket descriptor : 6
wait event...
timeout!
wait event...
timeout!
wait event...
[]

→ workspace ./client 127.0.0.1 50000
input message : aa
[RTT : 148.0 micro-seconds] echo message : aa
input message : ^C
→ workspace ./client 127.0.0.1 50000
input message : []

→ workspace ./client 127.0.0.1 50000
input message : bb
[RTT : 118.0 micro-seconds] echo message : bb
input message : ^C
→ workspace ./client 127.0.0.1 50000
input message : []

→ workspace ./client 127.0.0.1 50000
input message : cc
[RTT : 96.0 micro-seconds] echo message : cc
input message : ^C
→ workspace ./client 127.0.0.1 50000
input message : []
```

그리고 역시 아무런 이벤트 없이 10초가 지나면, 정상적으로 timeout이 되고 서버가 아직 살아 있음을 확인할 수 있다.