Week 4

Exercise 4
How does TCP layer react at a server host when a TCP SYN segment arrives at the server but the backlog queue of the listen socket is
full. How is this visible in the client application that is trying to connect the server?

If the servers backlog queue of the LISTEN socket is full the server fill simply ignore the SYN package and does not send a RST segment.
This has the effect that the client fill simply try to send it the SYN-segment again and only causes some delay while connecting.

If the RST-segment was sent the client application needs to handle the RST-segment.
Ignoring the SYN lets the TCP's retransmission protocol take over.

Exercise 6

a) When the client has connected to the server, kill the master (parent) server using an appropriate signal (e.g., SIGINT), but make sure that the TCP connection between the client and the server child process is still open (client and server are running). What happens in the client and what happens at the TCP layer?

The parent's child process keeps running till the client disconnects and the echoing works normally if the same client program is kept running. On the tcp-layer the after the 3-way handshake neither process haven't closed their connections so the child process keeps running.

b) After killing the master server, restart it immediately (keep the previous connection between the client and child server still open). What happens in the restarted master server? Why? How should you program the server to be prepared to this problem and thereby solve it?

The master throws errno: 98 Adress allready in use. Include flag SO_REUSEADDR as a socket option so the kernel will let you reuse the busy port if it's in TIME_WAIT state.

c) Now try the following: when the client has connected to the server, kill both the master and child server using an appropriate signal (e.g., SIGINT). What happens in the client and what happens at the TCP layer? Include appropriate error checking in the client.

Client doesn't react until you try to input something and write returns -1. Server sends FIN and client send ACK on the TCP-layer but client doesn't check the connection state.

d) After killing the servers as instructed above in c), restart the master server immediately. What happens now? Why? (Hint: look at the TCP state diagram on page 41 in Stevens book) How can you program the server to be prepared to this problem and thereby solve it?

The kernel keeps the connection open even after the processes are killed. Rebooting the server throws errno: 98 Adress allready in use. The server should close all sockets in a signal handler