Lernzielorientierte Fragen

1. Wie öffnet ein TCP-Server eine Verbindung?

socket()
bind()
listen() <= passive open
accept() <= accept connection
fork()
read()/write()

2. Wie öffnet ein TCP-Client eine Verbindung?

socket()
connect()
read()/write()

3. Was versteht man unter einem Port?

Ein Port ist gibt den jeweiligen Applikationsempfangskanal auf Layer 4 der OSI Models an. (TCP, UDP (oder RAW Socket))

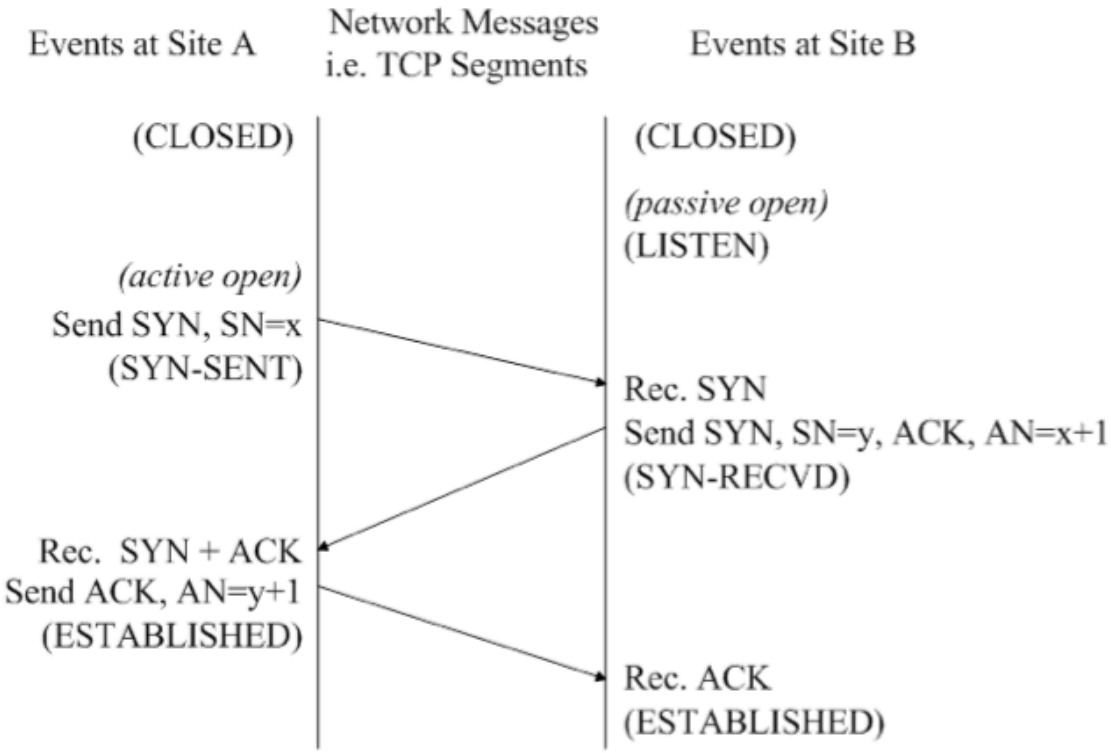
4. Was ist ein Socket?

abhängig des Kontexts:
IP:Port
Kombination aus ServerIP:Port / ClientIP:Port
Socket API
socket() Syscal

5. Wodurch ist eine TCP/IP Verbindung zwischen Client und Server eindeutig identifiziert?

Durch die beiden Sockets (IP:Port Kombination des Servers und Clients)

6. Wie funktioniert der 3 way handshake bei einem TCP-Verbindungsaufbau?



7. Was ist der Unterschied zwischen UDP und TCP?

TCP

- verbindungsorientiert
- sichere Übertragung (Acknowledgments) und Retransmission
- Sliding window
- Sorting
- Verbindungsaufbau/Abbruch

UDP

- verbindungslos
- keine sichere Übertragung dafür weniger Overhead
- vor allem im für zeitkritische Anwendungen (z.B.: VOIP)

8. * Wie hängt das OSI-7-Schichtenmodell mit dem Internet-Modell zusammen?

Beides sind Schichtenmodelle, welche die Netzwerkkommunikation organisieren sollen. Das OSI Modell dient hautpsächlich als Lehrmodell (vor allem Schicht 5,6 eher schwamig). Die weiteren Schichten können fast Ein-Zu-Eins aufeinander gemappt werden

9. Was versteht man unter einem iterative server?

Ist ein Server, welcher immer nur einen Client bedienen kann.

10. Wieso können zu einem iterativen Server mehrere TCP/IP-Verbindungen im Zustand ESTABLISHED sein?

Eine Verbindung wird als ESTABLISHED markiert, wenn der 3-Way-Handshake erfolgreich durchgeführt wurde. Dies wird vom System für die Applikation erledigt. Wenn nun zu einer bereits bestehenden Verbindung ein zweiter Client anfragt, kann die Verbindung vom System entgegengenommen und ESTABLISHED, aber keine Daten mit der Applikation ausgetauscht werden

11. Wer verwaltet die lokalen TCP bzw. UDP Portnummern?

Die lokalen Portnummern werden vom Betriebssystem vergeben.

12. Nach welchen Regeln werden die Portnummern vergeben?

- *Well known ports* (0 bis 1023) werden von der IANA reguliert.
- *Registered ports* (1024 bis 49151) werden von der IANA nicht reguliert, jedoch registriert und veröffentlicht (z.B. die X-Server-Ports 6000 bis 6063).
- *Dynamic or private ports* (49152 bis 65535) sind *ephemeral* und frei verfügbar.

13. Was versteht man unter einem raw socket?

Ein raw socket, ist eine Verbindung, welche direkt auf das IP Protokol aufsetzt, aber nicht z.B.: TCP oder UDP nutzt.

14. Wie nennt man einen UDP-Socket noch?

datagram socket

15. Was ist ein stream socket?

TCP-Socket

16. Was ist ein connected socket?

Es gibt ein vollständiges Socket-Paar (Server & Client Socket)

17. Was ist ein listening socket?

Es gibt nur einen Server Socket (passive open)

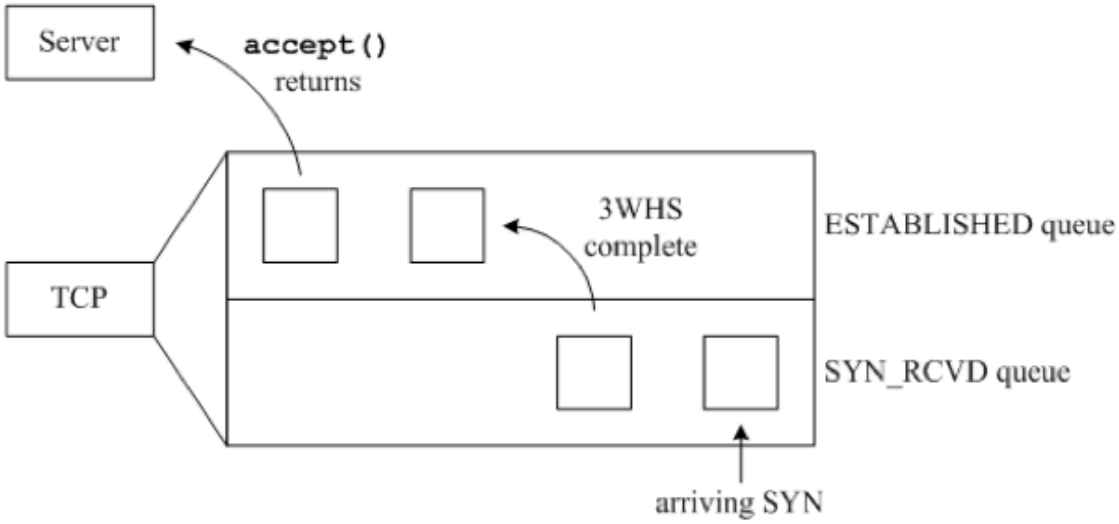
18. Warum verfügt der child process eines Forking-Servers sowohl über den listening socket als auch über den connected socket?

Es werden alle sockets bei einem fork dubliziert (d.h. sowohl der listening als auch der connected). Es sollten die entsprechenden nicht benötigten Sockets geschlossen werden

19. Welche Aussage hat der backlog Wert?

Maximale Länge der Warteschlange, welche die Connections vor dem accept() Syscall sichert (machmal auch die max. Länger der Queue für incomplete connection queue).

- *Completed connection queue* für Verbindungen im ESTABLISHED Zustand
- *Incomplete connection queue* für Verbindungen im SYN_RCVD Zustand



20. Manche Betriebssysteme haben anstelle des fork() einen spawn() SysCall. Worin liegt (meist) der Unterschied zwischen fork() und spawn()?

spawn: Kombination aus fork und exec; jedoch wird der aktuelle process space überschrieben (ausführen eines externen Programmes)

21. Welche Auswirkung kann die sliding window Fluss-Steuerung auf den Datenaustausch zwischen Client und Server haben? (Tipp: Denken Sie an die Eigenschaften der SysCalls read() und write())

Abhängig von der Größe des Sliding Windows können mehrere ACKs ausständig bleichen. Damit kann der Oberhead verringert werden.

22. * Was ist im Sinne von UNIX bzw. Linux ein socket descriptor?

socket descriptor sind im Endeffekt file descriptors (man kann read und write darauf anwenden).
Es handelt sich um einen Integer, welche als Index für einen Kernel-Level Datenstruktur fungiert. In dieser Datenstruktur sind alle notwendigen Infos gespeichert um vom richtigen "Objekt" zu lesen (z.B.: Datei, Socket)

23. * Unter welchen Bedingungen kehrt der read() SysCall zum Aufrufer zurück?

Erfolgreich x bytes gelesen (x wird returned)
On error (-1 wird returned)
Abbruch (z.B.: pipe brocken)

24. * Was bedeutet die Aussage „die Funktion accept() blockiert"?

Die Funktion accept() wartet so lange, bis eine neue Verbindung eintrifft. Soll heißen der Programmfluss wird so lange angehalten.

25. * Wie kann eine bestehende TCP-Verbindung „abreißen"?

Server oder Client fallen aus; das Netzwerk fällt aus

26. Welche Vorkehrungen sind im parent process notwendig, damit ein terminierter child process nicht in den Zombie-Zustand fällt?

Vaterprozess muss Status abfragen (waitpid)
oder Signalhandler (SIGCHILD)

Wenn SIGCHILD mit waitpid() warten bis alle Resourcen geschlossen => sonst zombie

Vorbereitende Fragen

siehe https://cis.technikum-wien.at/documents/bic/3/vcs/download/vcs_tcpip/ss6543.c

27. Was ist die prinzipielle Funktion des Servers?

Iterativer Server, welcher eine Verbindung auf Port 6543 öffnet.
Sobald eine Verbindung eintrifft wird eine Welcome Message gesendet.
Danach können die Befehle si, ci, hi und ex ausgeführt werden.
si: gibt die Server IP und den Port zurück
ci: gibt die Client IP und den Port zurück
hi: meldet “nice to meet you”
ex: beendet die Verbindung

28. Welche Kommandos unterstützt der Server?

siehe oben

29. Welche Werte kann `errno` annehmen, wenn `bind(2)` einen Fehler liefert?

EACCES: The address is protected, and the user is not the superuser.
EADDRINUSE: The given address is already in use.
EBADF: sockfd is not a valid descriptor.
EINVAL: The socket is already bound to an address.
ENOTSOCK: sockfd is a descriptor for a file, not a socket.

The following errors are specific to UNIX domain (AF_UNIX) sockets:
EACCES: Search permission is denied on a component of the path prefix. (See also path_resolution(7).)
EADDRNOTAVAIL: A nonexistent interface was requested or the requested address was not local.
EFAULT: addr points outside the user's accessible address space.
EINVAL: The addrlen is wrong, or the socket was not in the AF_UNIX family.
ELOOP: Too many symbolic links were encountered in resolving addr.
ENAMETOOLONG: addr is too long.
ENOENT: The file does not exist.
ENOMEM: Insufficient kernel memory was available.
ENOTDIR: A component of the path prefix is not a directory.
EROFS: The socket inode would reside on a read-only file system.

30. Welche Programmierfehler enthält der Server?

- kein Error Checking
- kein close des listening sockets
- lässt sich mit *gcc -Wall -Werror -Wextra -pedantic ss6543.c* nicht compilieren (weil warnings)
- keine shutdown Möglichkeit

compile using: gcc -Wall -Werror -Wextra -pedantic ss6543_fixed.c