

## 1. Begriffe

**BIOS** BIOS steht für Basic Input Output System und ist eine fest gespeicherte Software, die bei jedem Start des PCs ausgeführt wird. Sie ist für anfängliche Hardwaretests und Selbsttest verantwortlich (Memory check etc.) und initiiert die für das Booten des Betriebssystems nötigen Prozesse.

**Master Boot Record** Das ist der erste Sektor eines bootbaren Mediums, in dem die Partitionstabelle und der Bootloader abgespeichert sind. Das BIOS guckt sich den MBR an und lädt den Bootloader dann in den Speicher.

**Cron Job** Ein Cron Job bezeichnet unter einem UNIX System eine immer wiederkehrende Aufgabe, die zu einem bestimmten Zeitpunkt ausgeführt wird. Dabei steuert der Cron-Deamon die Ausführung der Jobs, dies können z.B. Backup-Programme o.ä. sein.

**Protocol Stack** Als Protocol Stack wird eine Ansammlung von verschiedenen Protokollen bzw. Protokollschichten bezeichnet. Und zwar sind die Schichten in der Art angeordnet, dass die Schichten aufeinander gestapelt sind und jedes Protokoll nur die Dienste der direkt darunter liegenden Schicht verwenden darf.

## 2. Protokolle des OSI-Schichtenmodells

**SMTP** Das ist das Simple Mail Transfer Protocol, welches logischerweise auf Schicht 7, der Anwendungsebene läuft. Und zwar ist es wie der Name schon sagt für die Übertragung von Emails zuständig, spezieller ist dieses Protokoll für das Versenden und Weiterleiten zuständig, nicht aber für den Empfang (das läuft dann über POP3 oder IMAP).

**DNS** Das so genannte Domain Name System. Die Hauptaufgabe des Protokolls ist die Umsetzung von Internetadressen in die zugehörige IP-Adresse. Demzufolge ist das Protokoll auch der 3. Schicht - dem Networklayer zuzuordnen, da es sich um ein Weiterleitungs- und Routingproblem handelt.

**SNMP** Das ist das Simple Network Management Protocol, es ist ein Netzwerkprotokoll, das Netzwerkelemente (z. B. Router, Server, Switches, Drucker, Computer usw.) von einer zentralen Station aus überwacht und steuert. Demzufolge ist es auch auf der Anwendungsschicht anzusiedeln.

**ICMP** Das so genannte Internet Control Message Protocol dient in Netzwerken zum Austausch von Informations- und Fehlermeldungen. Da es quasi zur Überwachung der Verbindungen dient und somit für Weiterleitungs- und Routingprobleme wichtig ist, ist es auf Schicht 3 (Network-Layer) einzuordnen.

**UDP** Das User Datagram Protocol ist auf Schicht 4, dem Transport-Layer anzusiedeln. Und zwar stellt UDP eine beidseitige Verbindung zwischen zwei Sendern bereit, die keine Rücksicht auf Verluste - im wahrsten Sinne des Wortes - nimmt.

**IP** Dies ist das Internet Protocol, welches dafür sorgt, dass Computer mittels IP-Adresse und Subnetmaske in identifizierbare, logische Gruppen unterteilt werden können, so dass es möglich ist, Computer in größeren Netzwerken zu adressieren. Demzufolge ist IP auf Schicht 3 (Networklayer) anzusiedeln, da es grundlegend für die Vermittlung und Zustellung zuständig ist.

**FTP** Das File Transfer Protocol ist ein auf der Anwendungsschicht 7 ansässiges Protokoll. Es stellt die Funktionalität zu Verfügung, dass ein Client mit einem Server Dateien austauschen kann.

### 3. Designprinzipien des Internets

1. Ein Prinzip ist, dass man das Internet so minimalistisch wie möglich gestalten will und die einzelnen Server autonom verwalten lassen. D.h. ein unabhängiger Betrieb des Netzwerkes soll ermöglicht werden, bei dem interne Veränderungen nicht notwendig werden, wenn man sich noch zu anderen Netzwerken verbindet.  
Bei einem Verstoß dieser Regeln könnte es passieren, dass verschiedene Netzwerke so konzipiert würden, dass sie nicht mehr miteinander kommunizieren könnten. Zudem könnte es passieren, dass bei Verstoß gegen die Autonomie Netzwerke, die zusammenhängen, auch gleichzeitig aufgrund der nun vorhandenen Abhängigkeiten ausfallen.
2. Bei dem Best-Effort Modell versucht das ganze Netzwerk so gut wie möglich die Daten vom Sender zum Empfänger zu übermitteln. Die sichere Kommunikation kann hierbei über wiederholtes Übertragen realisiert werden.  
Beim Verstoß gegen diese Designregel könnte die Zuverlässigkeit und Richtigkeit der Daten nicht mehr garantiert werden, z.B. wenn man das wiederholte und redundante Senden weglassen würde.
3. Bei dem Stateless intermediate Prinzip sollen die Zwischenpunkte wie Router keinen Status bezüglich einer durchgehenden Verbindung halten, ganz im Gegensatz zu typischen Telefonnetzwerken
4. Als letztes gibt es das Decentralized control Prinzip, bei dem wie der Name schon sagt kein globale bzw. zentralisierte Instanz, die alle Verbindungen der Netzwerke miteinander kontrolliert.  
Auch hier wäre bei Verletzung der Prinzipien denkbar, dass durch ein Fehlverhalten einer kontrollierenden Instanz das ganze Netzwerk lahmgelegt wird. Zudem müsste jeder neue Zugangspunkt dort erst autorisiert werden, was einen enormen Administrativen Aufwand mit sich bringen würde.

## 4. Systemdienste

Eine kleine Auflistung von Windows Diensten.

**Plug & Play** Der Plug & Play Dienst ermöglicht es, dass Hardwareänderungen erkannt und mit möglichst wenig Benutzeraktivität eingerichtet werden. Und da das Beenden dieses Prozesses nicht erlaubt ist, nehme ich an, dass der Betrieb nicht mehr so wirklich gewährleistet wäre. Vermutlich stürzt dann alles ab, weil kein Plug& Play-Gerät mehr laufen möchte (was heutzutage ja fast alles ist).

**Systemwiederherstellungsdienst** Dieser sorgt dafür, dass automatisch Systemwiederherstellungsfunktionen ausgeführt werden. Z.B. sorgt dieser Dienst dafür, dass in regelmäßigen Abständen die aktuellen Betriebssystemeinstellungen gesichert werden. Theoretisch beeinträchtigt das Beenden dieses Dienstes das Gesamtsystem nicht, es werden dann halt nicht mehr die Einstellungen und Änderungen gesichert.

**DHCP-Client** Der Dienst verwaltet die Netzwerkkonfigurationen und aktualisiert und registriert IP-Adressen und DNS-Server. Wenn der Dienst beendet wird, dann funktioniert die Netzwerkverbindung nicht mehr, wenn bei dieser die IP etc. per DHCP bezogen werden müssen. So kann sich dann also z.B. nicht mehr zum Internet verbunden werden.

**Server** Dieser Dienst ist verantwortlich dafür, dass im Netzwerk Datei- und Druckerfreigaben möglich sind. Bei Beendigung kann dann kein anderer Computer mehr auf die Freigaben zugreifen, was aber das Gesamtsystem nicht beeinträchtigt.

## 5. Fifty/Fifty

- *Ein Druckerdämon ist ein Virus, der sich auf das Stören der Funktionsweise eines Druckers spezialisiert hat.*  
Nein. Der Druckerdämon ist der unter UNIX laufende Systemdienst, der für das Drucken bzw. für die Bereitstellung der Druckfunktionalität verantwortlich ist. Z.B. verwaltet er die Druckerwarteschlangen usw.
- *Fehler in der Implementierung von Systemdiensten sind in ihren sicherheitsrelevanten Auswirkungen auf den jeweiligen Systemdienst begrenzt.*  
Nein, und zwar können Dienste, die für die Sicherheit des Gesamtsystems verantwortlich sind (wie z.B. Windows-Firewall oder Virens Scanner) durch schlechte und fehlerhafte Implementierungen das ganze System beeinträchtigen und gefährden.
- *Nach dem Socketaufbau ist es für die durch den Socket kommunizierenden Prozesse egal, ob sie auf dem selben oder auf unterschiedlichen Rechnern laufen.*

## 6. Dateioperationen

Als erstes *cp*.

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    FILE *dat, *cpy;
    char *name;
    int size = 0;
    if(fopen(argv[1], "rb") != NULL && argc >= 3) {
        dat = fopen(argv[1], "rb"); // zu kopierende Datei
        cpy = fopen(argv[2], "wb"); // zu schreibende Datei
        while(!feof(dat)) { //solange bei zu kopierenden noch nicht
            das Ende erreicht ist
                fread(stdin, 1, 1, dat); // lese 1 byte nach stdin ein
                fwrite(stdin, 1, 1, cpy); // schreibe 1byte von stdin nach
                    cpy
                size++; // zähle die Bytes
            }
        size--; // da einer zuviel gezählt wird, einen abziehen
        printf("%d Byte kopiert\n", size);
        fclose(dat); // Dateien schließen
        fclose(cpy);
    }
    else
        fputs("Fehler beim Öffnen oder zuwenig Parameter\n", stderr);
    return 0;
}
```

Und jetzt noch *mv*

```
#include <stdio.h>

int main(int argc, char* argv[]) {
    if(argc < 2)
        fputs("Fehler, zuwenig Argumente", stderr);
    rename(argv[1], argv[2]); // Datei wird umbenannt
    printf("\n");
    return 0;
}
```

Alternativ könnte man natürlich auch *cp* verwenden und dann noch die ursprüngliche Datei löschen, aber dabei würden dann die Metainformationen verändert werden.

## 7. Sockets

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#define BUFFER 1024
int main(int argc, char *argv[]) {
    if(argc == 3) {
        int s = socket(AF_INET, SOCK_STREAM, 0);
        int port;
        struct hostent *ip;
        struct sockaddr_in server;
        char erg[BUFFER];

        if(atoi(argv[2]) == 0) {
            fputs("Fehler beim Port, Parameter kann nicht in Zahl umgewandelt werden\n", stderr);
            return 1;
        }
        port = atoi(argv[2]); // wandelt String in Int um ->
                               Portnummer
        ip = gethostbyname(argv[1]); // wandelt Name in IP um
        server.sin_addr = *(struct in_addr*)ip->h_addr; // gibt
                                                         die IP an das Adress-Struct weiter
        server.sin_port = htons(port); // gibt den Port im
                                       richtigen Format an das Adress-Struct
        server.sin_family = AF_INET; // gibt den Typ der
                                      Verbindung an das Adress-Struct
        connect(s, &server, sizeof(server)); // Verbindung über s
                                              mit server und der Länge von der Server Adresse
        recv(s, erg, sizeof(erg)-1, 0); // auf s lauschen und in
                                         erg so viel wie Platz ist schreiben
        printf("%s", erg); // Ergebnis ausgeben
        close(s); // Verbindung schließen
    }
    else {
        fputs("Fehler, zu wenig Parameter\n", stderr);
        return 1;
    }
    return 0;
}
```