Übungen zur Vorlesung
BETRIEBSSYSTEME I
Studiengang Informatik
SS 03

AG Betriebssysteme FB3
Kirsten Berkenkötter, Jan Peleska

Ausgabe: 16.6.2003
Abgabe: 30.6.2003

Übungszettel 1

Hinweise

Die Abgabe kann entweder auf Papier oder als E-Mail an *kirsten@tzi.de* erfolgen. **Auf jeden Fall** sollten die C-Programme auch in elektronischer Form (als E-Mail-Attachment) abgegeben werden.

Bitte immer die Namen aller Gruppenmitglieder und die Gruppennummer angeben!

Aufgabe 1: UDP-Host

Programmieren Sie einen UDP-Host, der mit anderen Hosts über eine verbindungslose Punkt-zu-Punkt-Kommunikation Kontakt aufnehmen kann. Als Argumente sollen dem Programm der Port für den eigenen Socket sowie der Hostname und der Port für den anzusprechenden Socket übergeben werden. Für die verbindungslose Kommunikation werden die Funktionen *socket()*, *bind()*, *sendto()* und *recvfrom(),* sowie *setsockopt()* verwendet.

Mit Hilfe von *fork()* soll sich das Programm in einen Eltern- und einen Kindprozess spalten, wobei der erste für das Senden von Daten und der zweite für das Empfangen zuständig sein soll.

Die beiden Prozesse sollen sich über ein vorher angelegtes Shared Memory miteinander synchronisieren. Dieses soll durch ein Semaphor geschützt werden. Im Shared Memory sollen sich je ein Sende- und Empfangspuffer sowie benötigte Kontrollstrukturen befinden. Die beiden Puffer sollen Platz für mindestens zwei Datenpakete (*Puffergroesse* > *1*) bereitstellen.

Das Senden von Daten soll in zufälligen Abständen geschehen. Diese werden mit Hilfe von *srandom()/random()* ermittelt. Die maximale Datenlänge soll 4000 Bytes betragen, um die zulässige Größe für UDP-Pakete nicht zu überschreiten.

Die gesendeten und empfangenen Daten sollen zur Kontrolle auf dem Bildschirm ausgegeben werden.

Aufgabe 2: UDP-Protokoll

a) Implementieren Sie ein kleines Protokoll, um die sichere Übertragung von Daten per UDP für zwei der Hosts aus Aufgabe 1 zu gewährleisten. Folgende Messages sollen dazu verwendet werden:

CON_REQ	Connection Request, Verbindungsaufbau
CON_ACK	Connection Acknowledge, Verbindungsaufbau bestätigt
CON_REJECT	Connection Reject, Verbindungsaufbau zurückgewiesen
CON_DATAx	Connection Data x, Datenpaket x gesendet
CON_DATA_ACK	Connection Data Acknowledge, Datenpaket empfangen
CON_CLOSE	Connection Close, Verbindungsende
CON_CLOSE_ACK	Connection Close Acknowledge, Verbindungsende bestätigt

Um die Implementierung zu vereinfachen, soll zunächst jeweils nur ein Datenpaket gesendet werden. Die Datenpakete brauchen auch nicht mit Daten gefüllt werden, das Senden der entsprechenden Nachrichten genügt. Eine erfolgreiche Verbindung hat also folgendes Aussehen:

```
CON_REQ
                CON_ACK
CON_DATA1
                CON_DATA_ACK
CON_CLOSE
                CON_CLOSE_ACK
```

Es kann immer nur ein Kommunikationspartner eine Verbindung aufbauen, um Daten zu senden. Der andere Partner sendet währenddessen nur Quittungen. Nach Ende der Verbindung kann er selbst eine Verbindung aufbauen und Daten senden (wird durch die zufälligen Zeiten beim Senden bestimmt, s. Aufgabe 1).

Kann eine Verbindung nicht angenommen werden, wird CON_REJECT als Quittung geschickt. Danach muss ein neuer Verbindungsaufbau vorgenommen werden.

Um nicht auf jede Quittung einzeln warten zu müssen, können mehrere Nachrichten nacheinander geschickt werden. Die Anzahl der noch nicht quittierten Nachrichten darf aber *Puffergroesse-1* nicht überschreiten. In diesem Fall wird ein Flag *XOFF* gesetzt. Erst wenn das Senden wieder möglich ist, wird dieser auf *XON* zurückgesetzt. Nach zehn erfolglosen Sendeversuchen bei *XOFF* wird angenommen, dass keine Quittung mehr erfolgt, ein neuer Verbindungsaufbau muss vorgenommen werden.

Tritt ein Fehler auf (z.B. eine falsche Quittung, mehrere gleiche Nachrichten hintereinander, etc.) muss ein neuer Verbindungsaufbau vorgenommen werden.

b) Erweitern Sie das Programm, so dass mehrere Datenpakete hintereinander gesendet werden können. Die Anzahl der Pakete soll mit *srandom()/random()* ermittelt werden. Jedes Datenpaket erhält eine Nummer, also z.B. CON_DATA15, CON_DATA14, ..., CON_DATA1. Das Senden soll von der höchsten bis zur niedrigsten Nummer erfolgen. Als Bestätigung wird jeweils ein CON_DATA_ACK (ohne Nummer) gesendet. Es soll hier genügen, dass die Anzahl der Quittungen mit den gesendeten Paketen übereinstimmt.