Vorlesung Betriebssystemarchitektur WS 2002/03

Aufgabenblatt 6 vom 8. Januar 2003

(Vorstellung der Lösungen bei den Tutoren bis zum 31. Januar 2003)

Aufgabe 6.1: (55 Punkte)
Implementieren Sie in der Programmiersprache C ein Instant-Messaging-System, das folgende Eigenschaften aufweisen soll:

- Als Kommunikationsmedium soll IP mit UDP-Datagrammen (Portnummer 8777) benutzt werden.
- Die Datagramme dürfen maximal 200 Bytes groß sein.
- Das Instant-Messaging-System benutzt ein einfaches Protokoll, das mit zwei Datagramm-Typen arbeitet:
  - "online"-Meldungen, mit denen man anderen Teilnehmern mitteilt, dass man Text-Datagramme empfangen kann; in diesen Meldungen soll der eigene Name enthalten sein
  - die Text-Datagramme selbst
- Die beiden Datagramm-Typen werden durch die ersten vier Bytes unterschieden. Für "online"-Meldungen wird der Präfix "**`ONL:`**" benutzt, für Text-Meldungen "**`MSG:`**". Der Inhalt der Datagramme soll im ISO8859-1-Code codiert sein. Bespiele:
  - **`ONL:mdirska`** ist ein gültiges "online"-Datagramm mit 11 Bytes
  - **`MSG:Hallo!`** ist eine gültige Text-Meldung (10 Bytes)
- Das System besteht aus drei Teilen:
  - einem Empfangsprogramm, das die Datagramme empfängt und auswertet (Hinweise zur Implementation: socket, bind, struct sockaddr_in, recv, recvfrom)
  - einem Sendeprogramm, das bei Bedarf Meldungen verschicken kann (send, sendto)
  - einem "online"-Programm, das durch periodisches Senden von "online"-Meldungen in's lokale Subnetz Empfangsbereitschaft signalisiert (setsockopt, SO_BROADCAST) – bitte nicht mehr als 20 Broadcast-Datagramme pro Minute senden!
- Idealerweise sollte das Empfangsprogramm und das "online"-Programm in einem Prozess laufen. Am besten in einem Thread (Hinweis: select).
- Das Empfangsprogramm soll nicht jede empfangene "online"-Meldung ausgeben sondern lediglich mitteilen, wer gerade "online" ist. Wenn von einem Rechner 10 Sekunden keine "online"-Meldung empfange wurde, dann ist davon auszugehen, dass der Benutzer sein Empfangsprogramm beendet hat ("offline").

Schreiben Sie das Programm/die Programme so, dass dergleiche Quellcode sowohl unter Linux als auch unter Windows kompiliert werden kann (Hinweis: Präprozessor, **`#ifdef _WIN32`**). Bei Windows muss vor Benutzung der Socket-Routinen zunächst die Winsock-DLL initialisiert werden (WSAStartup). Beim Linken der EXE-Datei muss die Winsock-Bibliothek angegeben werden
(z.B. so: "**`cl programm.c ws2_32.lib`**").

Schicken Sie mindestens 24 Stunden vor dem Tutoriumstermin den Quellcode des Programms/der Programme als Attachment an `bs@hpi.uni-potsdam.de` mit dem Betreff
`AUFG6.1-UEBUNGSGRUPPE=`<Ihre Übungsgruppennummer> .

Aufgabe 6.2: (45 Punkte)
Gegeben sei ein Fileserver, der folgende Schnittstelle zum Zugriff auf die dort abgelegten Dateien bietet:

- Es gibt einen RPC (remote procedure call) "read", mit dem man den Inhalt von Dateien auslesen kann. Übergabeparameter:
    - Datei-Nummer (Inode-Nummer), eine Datei wird auf dem Server nur durch diese Zahl identifiziert,
    - die Anzahl der Bytes, die gelesen werden soll
    - und ein Anfangspunkt in der Datei, gemessen in Bytes vom Anfang der Datei (Offset).

  Zurückgegeben werden maximal soviele Bytes, wie angefordert wurden. Weniger, wenn vorher das Ende der Datei erreicht wurde. Es werden Null Bytes zurückgegeben, wenn der angegebene Offset größer oder gleich der Dateigröße ist.
- Es gibt Dateien, die Verzeichnis-Informationen enthalten. Diese Dateien haben einen festgelegten Aufbau: sie enthalten mehrere Verzeichnis-Einträge der festen Länge 32 Byte, die folgenden Inhalt haben: die ersten 4 Byte ergeben zusammengesetzt eine 32-Bit-Zahl (MSB first - most significant byte first). Dies ist die Datei-Nummer (Inode-Nummer) des Verzeichnis-Eintrages. Die nächsten 27 Byte enthalten einen Dateinamen im ASCII-Code. Namen kürzer als 27 Byte werden mit binären Nullen aufgefüllt. Das letzte Byte ist entweder binär 0 oder 1. Eine 0 bedeutet, dass dieser Eintrag eine normale Datei beschreibt. Eine 1 bedeutet, dass die angegebene Datei wiederum eine Verzeichnis-Datei ist. So lässt sich ein hierarchischer Verzeichnis-Baum aufbauen.
- Die Datei mit der Nummer 1 ist eine Verzeichnis-Datei und bildet die Wurzel des Dateibaumes.
- Jede Verzeichnis-Datei enthält mindestens zwei Einträge:
    - "." mit der Datei-Nummer der eigenen Verzeichnis-Datei (zeigt also auf sich selbst).
    - ".." mit der Datei-Nummer der Verzeichnis-Datei, die in der Hierarchie eine Ebene tiefer liegt (näher an der Wurzel).
- Im Wurzel-Verzeichnis zeigen die Einträge "." und ".." auf die gleiche Datei-Nummer 1.

Sie sollen ein Programm konzipieren, dass mit diesem Fileserver kommuniziert. Dazu benötigt dieses Programm u.a. eine Variable, die das momentane Arbeitsverzeichnis (current working directory) beinhaltet. Diese Variable speichert in diesem Fall die Datei-Nummer der aktuellen Verzeichnis-Datei. Diese Variable sei mit 1 initialisiert (Wurzel-Verzeichnis).

Denken Sie sich Algorithmen für die Funktionen "cd" (change working directory) und "pwd" (print working directory) aus. "cd" soll einen Pfad-Namen als Übergabe-Parameter bekommen, bei dem die Pfad-Komponenten mit "/" getrennt sind. "pwd" soll auf der Standard-Ausgabe den kompletten Pfad-Namen des aktuellen Verzeichnisses ausgeben. Erstellen Sie jeweils ein Ablaufdiagramm (UML oder FMC) für folgende Funktionsaufrufe: "cd ./usr/include/sys" und anschließendes "pwd".

Bringen Sie die beiden Diagramme in Papierform zum Tutoriumstermin mit und geben Sie sie bei Ihrem Tutor ab.