Übungsaufgabe #2: Kommunikations-Basissystem: BS-Abstraktionen

4.05.2005

Ziel der Aufgabe ist es ein Basissystem zu erstellt, das eine Systemabstraktion mit einer einheitlichen, einfachen Schnittstelle auf verschiedene Basismechnismen des Betriebssystem bietet (Nachrichtenkommunikation mit UDP oder serieller Kommunikation; Semaphore zur Synchronisation; einfache Thread-Verwaltung). Diese Basissystem ist sowohl für Unix (Linux/Solaris) als auch für den RCX und den RCX-Simulator zu entwickeln.

Das Kommunikationssystem ist stets auf eine objektorientierte Abstraktion mit nachfolgend beschriebenen Elementen abzubilden:

- *Initialisierung:* Der Konstruktor initialisiert das Kommunikationsobjekt. Er kann ein Adressobjekt (`Address`) als Parameter erhalten, der dann festlegt, auf welcher lokalen Adresse Daten empfangen werden können (z.B. IP-Adresse und Port für UDP oder Devicename bei der Kommunikation mit einem RCX). Wird keine lokale Adresse angegeben, wählt (soweit möglich) das System selbst eine. Mit der Methode `getLocalAddress` kann diese Adresse, unter der das Kommunikationssystem von aussen erreichbar ist, abgefragt werden. Als Adresse ist im Falle von UDP eine Klasse `InetAddress` (erbt von `Address`) zu verwenden, die eine `struct sockaddr_in addr` enthält. Eine Initialisierung über Hostname und Portnummer ist vorzusehen. Im Falle der seriellen Kommunikation mit dem RCX soll entsprechend eine `RCXAddress` Klasse entworfen werden.
- *Senden von Nachrichten:* Zum Senden von Nachrichten wird vom System eine Methode `void send(Address *addr, Buffer *buf, int len)` angeboten. Dabei soll sich die Methode asynchron puffernd verhalten. Die Daten im Puffer werden also bei Aufruf der `send`-Methode übernommen und vielleicht erst zu einem spätern Zeitpunkt versendet. Die Methode kehrt also sofort zurück und der Puffer kann wieder überschrieben werden. Konkret muss das Kommunikationssystem dafür eigene Puffer erzeugen und verwalten welche die Daten zwischenspeichern. Dabei soll die Anzahl der Puffer begrenzt sein, ihre Größe kann jedoch variieren. (*Anmerkung:* Zu Testzwecken ist dies in der Unix-Variante zu implementieren, zwingend nötig ist es jedoch nur für den RCX und den RCX-Simulator.)
- *Empfangen von Nachrichten:* Beim Empfangen von Nachrichten ist ein aktive (wartende) Methode `receive(Address *src, Buffer *buf)` bereitzustellen. Diese blockiert, bis eine Nachricht empfangen wurde.

Das Kommunikationssystem soll jeweils in einer Implementierung für UDP (Unix to Unix), UDP-Sim (Unix to RCX-Simulator) und serieller Kommunikation (Unix to RCX) bereitstellen. Ein mögliches Interface könnte also wie folgt aussehen:

```
class CommunicationSystem {
 public:
    CommunicationSystem(Address addr);
    virtual Address getMyAddress();
    virtual void send(Address dest, Buffer *message, int len);
    virtual bool receive(Address *src, Buffer *buffer);
};
class Buffer { public: char *buffer; int len; };
```

Als zweites ist eine zählende Semaphore bereitzustellen mit folgender Schnittstelle. Der Konstruktor erhält den initialen Wert des Zählers:

```
class Semaphor{
  public:
     Semaphor(int startvalue);
     void P(void);
     void V(void);
};
```

Als drittes ist eine einfache Thread-Verwaltung bereitzustellen, die intern direkt auf entsprechende Funktionen der pthread-Bibliothek (Unix) bzw. der entsprechenden RCX-Funktion abgebildet werden können. Der Konstruktor von Thread erzeugt einen neuen Thread, in welchem die Methode `run` der übergebenen Klasse (welche `Runnable` implementieren muss) ausführt. Falls in den folgenden Aufgaben weitere Thread-Methoden benötigt werden, so kann diese Klasse erweitert werden. Im Hinblick auf leichte Portierbarkeit ist die Schnittstelle dabei aber auf eine Minimallösung zu beschränken.

```
class Runnable { virtual void run() = 0; };
class Thread {
  private: ...
  public:
     Thread(Runnable *run);
};
```

a) Implementiere die beschriebene Funktionalität mit Unix. Es sollte auf Portabilität geachtet werden. (z.B. sowohl mit PC/Linux als auch mit Sun/Solaris testen!)

b) Implementiere die beschriebene Funktionalität für RCX und den RCX-Simulator.

Achtung: Auch wenn es in der Pfingstwoche keine Übung gibt wird eine neue Aufgabe gestellt!

Abgabe: bis 23.05.2005 16:00 Uhr
Alle benötigten Dateien sind im Verzeichnis /proj/i4vs/*loginname*/aufgabe2/ abzulegen und mit dem `abgabe` Programm abzugeben.