Übungsaufgabe #2: Kommunikations-Basissystem: BS-Abstraktionen

07.05.2007

Ziel der Aufgabe ist es ein Basissystem zu erstellen, welches eine Systemabstraktion mit einer einheitlichen, einfachen Schnittstelle auf verschiedene Basismechanismen des Betriebssystems bietet (Nachrichtenkommunikation mit UDP; Semaphore zur Synchronisation; einfache Thread-Verwaltung). Dieses Basissystem ist für Unix (Linux/Solaris) 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). 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.
- *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 späteren 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.
- *Empfangen von Nachrichten:* Beim Empfangen von Nachrichten ist eine aktive (wartende) Methode `receive(Address *src, Buffer *buf)` bereitzustellen. Diese blockiert, bis eine Nachricht empfangen wurde.

Das Kommunikationssystem soll in einer Implementierung für UDP (Unix to Unix) bereitstehen. Ein mögliches Interface könnte also wie folgt aussehen:

```
class CommunicationSystem {
public:
    virtual void send(Address dest, Buffer *message, int len);
    virtual int 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 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!)

Abgabe: bis 22.05.2007 12:00 Uhr

Alle benötigten Dateien sind im Verzeichnis *fax* des SVN-Repositories abzulegen.