Übungen zu Vorlesung
Betriebssysteme II
Sommersemester 1997

AG BS
Bettina Buth
Jan Peleska

Serie 1

Thread Programmierung

Aufgabe 1: Parallel-Server – Implementierung mit Solaris User Threads

Implementieren Sie die Grundstruktur für ein Client-Server System entsprechend folgender Beschreibung:

Kommunikation: über Berkley Sockets mit connection-oriented protocol und TCP/IP.

Client: Fordert nach einem Benutzerbefehl vom Server eine Ressource an. Das Betriebsmittel wird einfach durch einen eindeutigen Identifikator modelliert. Bei Erhalt des Betriebsmittels wird der Identifikator auf dem Bildschirm dargestellt. Nach einem weiteren Benutzerbefehl teilt der Client dem Server mit, daß das Betriebsmittel nicht mehr benötigt wird. Danach wartet der Client auf den nächsten Benutzerbefehl und beginnt dann mit der nächsten Betriebsmittelanforderung.

Server: Soll als *parallel server* implementiert werden: Ein ständig laufender Thread akzeptiert den Verbindungsaufbau durch die Clients und erzeugt dann dynamisch einen neuen Thread, der mit dem Client die Betriebsmittel-Vergabe/-Rückgabe verwaltet. Die Ressourcen werden in einem Stack der Maximaltiefe $n = 5$ verwaltet: der erzeugte Thread nimmt das oberste Element (das nur aus einer Betriebsmittel-Id besteht!) vom Stack und übergibt es an den Client. Ist der Stack leer, so sind keine Betriebsmittel mehr vorhanden. Der den Client bedienende Thread muß dann solange warten bis wieder Betriebsmittel verfügbar sind. Nach Rückgabe des Betriebsmittels legt der zuständige Thread die entsprechende Id oben auf den Stack und terminiert. Der Stack soll durch ein Array mit Counter implementiert werden das im gemeinsamen Adreßbereich der Threads liegt. Der Zugriff auf den Stack soll durch zwei Semaphoren gesteuert werden:

- `MUTEX` garantiert wechselseitigen Ausschluß beim Zugriff auf die Stackstruktur (binäre Semaphore),

- `NOTEMPTY` ist eine n-wertige Semaphore, die anzeigt,, ob noch Elemente im Stack vorhanden sind.

a) Führen Sie eine Implementierung mit Solaris User Threads durch (in C oder C++).

b) Beschreiben Sie Möglichkeiten, den Server durch explizite Erzeugung von Solaris LWPs zu optimieren.

Manual Pages mit relevanter Informationen:

- `threads(3T)`
- `_lwp_create(2)`

Aufgabe 2: Parallel-Server – Implementierung mit POSIX Threads unter SGI

Portieren Sie Ihre Lösung von Aufgabe 1 auf SGI unter IRIX mit POSIX Threads.

Aufgabe 3: POSIX Threads für LINUX

Entwerfen Sie eine Erweiterung des LINUX-Kernels, welche die volle Funktionalität der POSIX-Threads unterstützt. Beschreiben Sie, welche IPC-Library und welche Kernel-Komponenten Sie hierzu ändern würden.

Abgabe: Do, 29.05.97, in der Vorlesung