1 Aufgabe 7

- Parallelisieren Sie den Raytracer aus Aufgabe 1 mit Hilfe von Pthreads.
  - ▲ Unterteilen Sie hierzu das Bild in einzelne Streifen und lassen sie die Streifen von eigenen Threads berechnen.
  - ▲ Vergessen sie nicht, in Ihr Program den Aufruf von `thr_setconcurrency(8)` einzubauen, um auf der `cssun` mit allen 8 Prozessoren rechnen zu können!
  - ▲ Warum ist die einfache Aufteilung in 8 Streifen nicht optimal? Die Rechenzeit verkürzt sich, wenn mehr Streifen und damit (Userlevel-)Threads eingesetzt werden. Was ist die optimale Anzahl von Threads?

© Michael Schröder – a7.doc: 26/3/98

2 Aufgabe 8

- In dieser Aufgabe soll ein HTTP Server parallelisiert werden. Der serielle Source findet sich in `~i481000/src/tinyhttp`
- Zum Testen kann der WWW Baum des RRZE verwendet werden. Ein Aufruf wäre:
  `./tinyhttp 2345 /proj/websource`
  Danach könnte zum Beispiel über die URL
  `http://cssun.rrze:2345/docs/RRZE/Stoiber.html`
  auf den Server zugegriffen werden können.
- Da der Server zu langsam arbeitet, sollen die einzelnen Requests parallel von Threads abgearbeitet werden.

© Michael Schröder – a7.doc: 26/3/98

3 cont.

- Zur Implementierung soll das “Pool of Threads” Modell verwendet werden. Hierbei erzeugt der Server eine feste Anzahl von Clientthreads, die sich alle an einer Requestqueue blockieren. Der Serverthread nimmt neue Aufträge entgegen und stellt sie in die Queue, wo sie von den Clientthreads abgearbeitet werden. Dieses Vorgehen hat folgende Vorteile:
  - ▲ Lastbegrenzung: Die Anzahl der Threads ist fest vorgegeben.
  - ▲ Geschwindigkeit: Ein (langsamer) Aufruf von `pthread_create` entfällt.

© Michael Schröder – a7.doc: 26/3/98

4 cont.

- Implementieren Sie zuerst die Requestqueue. Sie benötigen dazu einen Mutex und eine Condition Variable, an der die Clients sich schlafen legen können, falls die Queue leer ist.
- Ein Request soll folgendermaßen aussehen:

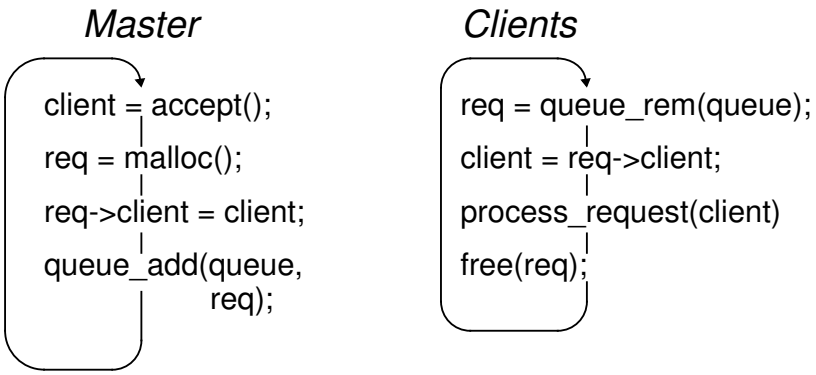
```
struct request {
    struct request *next; /* Verkettung */
    int client;           /* Descriptor */
};
```

- Über `queue_add(queue, request)` soll ein Request in die Queue eingehängt werden können und mit `queue_rem(queue)` soll der am frühesten eingehängte Request entfernt werden.

© Michael Schröder – a7.doc: 26/3/98

5 cont.

- Diagramm:



© Michael Schröder – a7.doc: 26/3/98

6 Aufgabe 9

- In `~i481000/src/life` befindet sich eine Implementierung von “Conway’s Game of Life”. Bei dieser Simulation ist das Überleben einer Zelle abhängig von der Anzahl der Nachbarzellen:
  - Ist die Anzahl < 2 stirbt die Zelle an Einsamkeit,
  - Ist die Anzahl genau 2, überlebt die Zelle,
  - Bei genau 3 Nachbarn wird eine neue Zelle erzeugt,
  - Eine Überpopulation von mehr als 3 Nachbarn führt wieder zum Tod der Zelle.
- Das Programm fungiert als kleiner WWW-Server. Das aktuelle Bild der Zellen wird über den *Server-Push* Mechanismus ausgegeben.

© Michael Schröder – a7.doc: 26/3/98

7 cont.

- Ändern Sie das Programm so, daß die Stadien der Population von vier Threads berechnet werden (Aufteilung in Streifen). Achten Sie darauf, daß die Randbereiche der Streifen richtig bearbeitet werden (Überschneidungen!).
- Am Ende der Zyklen müssen sich die Threads untereinander synchronisieren (Barriere). Implementieren sie dazu einen allgemeinen Barrierentyp.
  `barrier_init(struct barrier *b, int numthr)` soll die Barriere initialisiert werden, mit
  `barrier(struct barrier *b)` kann dann auf die anderen Threads gewartet werden.

© Michael Schröder – a7.doc: 26/3/98

8 cont.

- Im Moment kann das Bild der Population nur auf einem Browser dargestellt werden. Ändern sie den Code, so daß mehrere Browser bedient werden können. Erzeugen sie dazu für jede Anfrage an den Server einen eigenen Thread. Entkoppeln Sie ihn mit `pthread_detach` vom Hauptthread.
- Im Vollausbau sind also folgende Threads aktiv:
  - 1 Hauptthread, der Serveranfragen entgegennimmt.
  - n WWW-Clientthreads, welche die aktuellen Daten zu den Browsern schicken.
  - 4 Threads für die Berechnung der nächsten Population.
- Vergessen sie nicht den `thr_setconcurrency(8)` Aufruf!

© Michael Schröder – a7.doc: 26/3/98