Aufgabe 4:

msgserver (12 Punkte)

Schreiben Sie einen einfachen auf pthreads basierenden Nachrichtenserver **msgserver**. Dieser kann kurze Textnachrichten (max 2048 Zeichen) empfangen oder ausgeben.

a) Ringpuffer und Semaphore

Schreiben ein zunächst in der Datei **bbuffer.c** einen Ringpuffer der bis zu BUFFSIZE (=128) Einträge vom Typ int (d.h. Dateideskriptoren) aufnehmen kann. Die Schnittstelle soll den Vorgaben in **bbuffer.h** genügen. Beachten Sie, dass der Puffer so konzipiert werden soll, dass lesen und schreiben gleichermaßen aus mehreren konkurrierenden Threads möglich ist. Zur Synchronisation dienen Semaphore, die in **sem.c** implementiert werden und den Vorgaben aus **sem.h** entsprechen (**pthread_cond_wait(3)**, **pthread_cond_broadcast(3)**, **pthread_mutex_{, un}lock(3)**).

b) Message-Server

Der Server nimmt auf dem übergebenen Port Verbindungen entgegen und trägt die entsprechenden Dateideskriptoren in einen Ringpuffer ein. An diesem warten Threads (Anzahl bei Programmstart angegeben), die die eigentliche Behandlung der Anfragen übernehmen. Wurde eine Anfrage bearbeitet stehen die Threads für weitere Anfragen bereit. Nachrichten werden in eine Liste eingetragen oder aus ihr gelesen (Array oder verkettete Liste), die von allen Threads gemeinsam genutzt wird (Achtung Synchronisation nötig). Dabei wird folgendes Protokoll genutzt:

`GET` *n*`\n`	die n neuesten Nachrichten als Text (oder weniger, falls nicht vorhanden)
`PUT\n`	ok oder fail
Nachricht`\n`	

Pro Verbindung wird nur ein Befehl übertragen. Die Liste wird nie geleert

c) Lokale Nachrichtenlisten

Da sich die Threads beim auslesen der Listen bei GET sehr lange gegenseitig blockieren (nahezu über die gesamte Arbeitszeit hinweg), soll nun für jeden Thread eine eigene Liste eingeführt werden. Beim Empfangen der Nachrichten werden diese weiterhin in eine gemeinsame Liste gespeichert. Werden die Nachrichten aber ausgegeben, wird auf die lokale threadeigene Liste zugegriffen. Die Threads überprüfen bei jedem GET ob neue Nachrichten vorliegen (d.h. die gemeinsame Liste länger ist als die eigene) und übernehmen gegebenenfalls die neuen Nachrichten. Empfangene Nachrichten sollen bei PUT nicht direkt in die Liste geschrieben werden, sondern in einen Puffer, der – abhängig von Ihrer Implementierung – in die Liste kopiert oder eingehängt wird.

Allgemeine Hinweise

- Abgabe bis 17.07.2009 12:00
- es muss mit -pthread kompiliert werden
- als Client können Sie z.B. nc nutzen