Threads

Dinh-Phuong Duong - Markus Stuber

Erste lösung: Reader bevorzugt:
Wenn das Programm läuft und durch die Hilfe von mutex.lock() kritische Funktionen geschützt sind wobei diese bis sie wieder durch mutex.unlock() freigegeben werden Ausgeführt werden können.
Vorteil:
Sobald ein Reader in der Queue ist wird dieser einem Writer bevorzugt.
Durchschnittliche read time ist geringer als bei Writer bevorzugt.
Nachteil:
Das Problem wenn man den Reader immer dem Writer bevorzugt ist dass wenn es zuviele Reader Threads pro Writer Thread gibt d.h. Writer Threads werden einfach übergangen und es kommt nicht zu ihrem Aufruf, da immer ein oder mehrere Reader Threads mit Vorrang in der Warteliste stehen.

Erste Lösung: Writer bevorzugt:
Vorteil:
Hierbei wird der Writer bevorzugt es gibt zwar kleine Unterschiede.
Durchschnittliche writetime ist geringer als bei Reader bevorzugt.
Nachteil:
Jedoch kann auch hier jemand verhungern und zwar der Reader, wenn es zuviele Writer Threads gibt.
Ab und zu ist die Durchschnittliche read time ist ein wenig höher als bei Reader bevorzugt. Kann jedoch auch niedriger sein.

Zweite Lösung: Faire Behandlung:
Da in den Ersten Lösungen es jeweils zu Problemen kommen kann (starvation // verhungern) d.h. Wenn der Reader bevorzugt wird können die Writer eventuell verhungern und wenn Writer bevorzugt werden können die Reader verhungern.
Um dies zu umgehen, muss dafür gesorgt werden dass kein Reader oder Writer verhungert und oder vergessen wird.
Vorteil:
Niemand verhungert
Hat eine Durchschnittlich sehr geringe read und write time.
Die Durchschnittliche write time ist immer kleiner als bei Reader bevorzugt und ab und zu sogar geringer als bei Writer bevorzugt
Nachteil:
Die Durchschnittliche write time kann höher sein als bei Writer bevorzugt.
Macht eventuell weniger aufrufe overall depending on sleeps