Nils Petereit, Krzysztof Lagowski

nichtsequentiellen und verteilte Programmierung
Übungsblatt 3

Prof. Dr. Claudia Müller-Birn, Barry Linnert
Tutorium 01 (Alexander Rudolph)

Aufgabe 2:

Mutex Lösung:

Korrektheit: Der Code verwendet zwei Mutex-Locks, um den Zugriff auf den kritischen Abschnitt zu synchronisieren. Dadurch kann immer nur ein Thread die Brücke betreten, was Kollisionen verhindert und die Korrektheit gewährleistet.

Einfachheit der Implementierung: Die erste Mutex Lösung ist relativ einfach zu implementieren und zu verstehen. Sie verwendet bekannte Synchronisierungsmechanismen (pthread_mutex_lock und pthread_mutex_unlock), um die gegenseitige Ausgrenzung zu gewährleisten.

nicht hardware untersützte Lösung:

Korrektheit: Die Implementierung verwendet den Peterson-Lock-Algorithmus, um den gegenseitigen Ausschluss zu erreichen. Der Algorithmus stellt sicher, dass nur ein Thread gleichzeitig den kritischen Abschnitt betreten kann. Durch die Verwendung der Variablen level, last und lock im Algorithmus wird sicher gemacht, dass nur der Thread mit einer höheren Priorität den kritischen Abschnitt betreten kann. Dies gewährleistet die Korrektheit.

Einfachheit der Implementierung: Implementierung des Peterson-Lock-Algorithmus erfordert die Verwendung von zusätzlichen Variablen wie level, last und _lock. Diese Variablen müssen korrekt initialisiert und aktualisiert werden, um den Algorithmus korrekt zu implementieren. Das Verständnis und die korrekte Handhabung dieser Variablen erfordert möglicherweise ein tieferes Verständnis des Algorithmus.

Nils Petereit, Krzysztof Lagowski

Aufgabe 3: