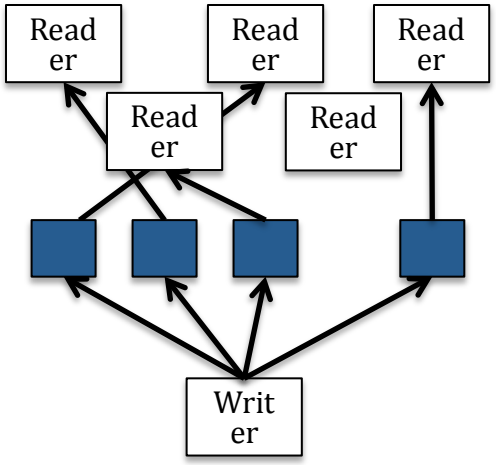
Vorlesung Betriebssysteme
Wintersemester 2017/2018

Prof. Dr. Lars-Olof Burchard
Hochschule Darmstadt

4. Praktikumsaufgabe (Abstimmungsprobleme)

Implementieren Sie mithilfe von POSIX-Threads, Mutex und Semaphoren unter Linux das folgende Problem (*Reader/Writer*):



1. Eine Menge von *Reader* und *Writer* Threads soll auf einen Datenspeicher (Array oder Vektor mit Strings oder Zahlen als Inhalt) mit konfigurierbarer Anzahl Plätze zugreifen.
2. Die Anzahl der *Reader*- und *Writer*-Threads sowie die Anzahl der Plätze in dem Array soll beliebig konfigurierbar sein. *Reader* und *Writer* sollen periodisch (in einer Schleife) auf die Datenstruktur zugreifen.
3. Ein *Writer* soll immer *alle* Plätze des Arrays gleichzeitig aktualisieren und *jeden* Platz mit einer Zufallszahl oder einem Zufallsstring belegen.
4. Ein *Reader* soll dabei periodisch jeweils an einer zufällig gewählten Stelle des Arrays lesen und die gelesenen Daten sowie den Index des gelesenen Elements auf der Standardausgabe ausgeben.

5. *Reader*-Threads können parallel lesen, d.h. mehrere *Reader* können und sollen (!) gleichzeitig auf den Array zugreifen. Das bedeutet, wenn ein *Reader* aktiv ist, können weitere *Reader* von *anderen* Elementen aus dem Array lesen. Wenn mehr *Reader* als Plätze in dem Array vorhanden sind, müssen manche Reader warten.
6. Ein *Writer* Thread kann immer nur exklusiv auf den Array zugreifen, d.h. zum Schreiben muss der Array für alle anderen *Reader* und *Writer* Threads gesperrt werden. Wenn *Reader* aktiv sind, muss ein *Writer* warten.

Hinweise

- Implementieren Sie *Reader* und *Writer* durch eigene Threads, d.h. es soll eine Threadfunktion für *Reader* und eine Threadfunktion für *Writer* implementiert werden, und dann jeweils die festgelegte Anzahl von Threads gestartet werden.
- Beobachten Sie, was passiert, wenn Sie Ihr Programm starten. Wie häufig können *Reader*, wie häufig können *Writer* auf die Datenstruktur zugreifen?
- Sorgen Sie durch geeignete Programmierung dafür, dass *Reader* und *Writer* nicht beliebig lange warten müssen, bevor Sie auf die Datenstruktur zugreifen können.
- Testen Sie Ihre Implementierung geeignet, d.h. untersuchen Sie die Auswirkungen von unterschiedlicher Anzahl *Reader*- und *Writer*-Threads sowie Anzahl Plätze und testen Sie dabei auch unterschiedliche Dauer der Periode der Lese- bzw. Schreibvorgänge (z.B. durch Verzögerung des Lesens/Schreibens mittels **`sleep`** bzw. durch Weglassen der Verzögerung).