

Nichtsequentielle und verteilte Programmierung Übung 02

Dozent: Prof. Dr. Claudia Müller-Birn, Barry Linnert

Tutor: Günes, Mehmed Hilmi Kerim

Studenten: Manuel Zschaebitz

3. Mai 2024

1 Kritischer Abschnitt in C

siehe git, alle Ausgaben sind in test*.txt es kommt zu vielen Unfällen Die Unfälle finden fast bei jedem crossing statt. Wobei unklar ist ob diese immer gleich crossen oder ob es einfach zu einer racecondition kommt, da die wartezeiten so kurz sind das der zugriff fast immer auf false bleibt oder gesetzt wird. zudem können beide gleichzeitig draufzugreifen, also ist es entweder false oder wird false gesetzt beim befahren.

edit: ok ich habe irgendwas verändert und habe 0 collisions jetzt ??? sorry ich hab irgendn fehler gerade in den globalen nicht globalen variablen and cant find it

ok ich habs kaputt gefixed lol, vor ein paar commits hats geklappt jetzt kurz vor abgabe habs ich leider zerstört :D

2 Sicherung des kritischen Abschnitts in C

siehe git, es kommt immer noch zu einer menge Unfällen, augenscheinlich bringt das lock aus der Vorlesung kaum etwas, wobei ich vielleicht auch hier einen Fehler habe den ich nicht rechtzeitig gefunden habe :D

correction: ich hab kurz vor abgabe den Fehler im Code gefunden, es kommt zu keinen collisions

2nd correction: jetzt kommt es wieder zu collisions ahahahahahahah

3 Erweiterung der Anzahl der Threads in C

Hier wirds spannend, denn ich hatte richtig Schwierigkeiten und habe einen SIGBUS error bekommen, musste ich auch lernen. Anscheinend war meine Zuweisung der threads Variable ungültig was dazu führte, anyhow mit ChatGPT und atol konnte es gelöst werden.

Ich habe für die 3 ein anderes lock Verfahren genutzt was in pthread vorhanden war. Weil ich aus der 2 sah das es nicht wirkte. Gerade lese ich erst das wir ein Verfahren (lock!?) aus der VL hätten nutzen sollen.

Huppala :D

4 Bewertung der Ansätze

Simulation 1 & 2 unterscheiden sich eigentlich nicht von den Unfall Zahlen her, es scheint das das implementierte lock fehlerhaft oder aber unzureichend funktioniert. Da sich alles im Millisekunden bereich abspielt ist es gut möglich das beide threads aufeinander warten und dementsprechend im gleichen moment Zugriff bekommen.

Correction ich hab den Fehler gefunden, jetzt ist es aber zu spät um ne neue lock Lösung für 3 zu bauen :D sorry

Also Simulation 1 versucht fast bei jeder überquerung unfälle, was vermutlich daran liegt das die Zeit einfach zu kurz ist in Millisekunden und ich durch den Machineoverhead immer zu kollisionen komme auch da zwei Threads auf die gleiche Bool Variable zugreifen.