Fabian Hupfeld (35523329) und Konstantin Stitz (35532242)

Übungsaufgaben 8

Aufgabe 1.)

a.)

(1)

Bei Testalgorithmus 1 wird eine statische Zählvariable verwendet (m_nextTurn) die durchgezählt wird und immer, wenn diese (modulo der Philosophenanzahl) gleich der jeweiligen eigenen ID ist, darf der entsprechende Philosoph essen, indem er erst beide Gabeln nimmt und dann wieder ablegt. Der Nachteil dieses Algorithmus ist, dass immer nur ein Philosoph gleichzeitig essen kann und das warten auf das Essen mit einem ineffizienten busy waiting umgesetzt ist.

(2)

Nein der Algorithmus ist nicht fair. Zwar ist es so dass jeder Philosoph der Reihenfolge nach drankommt, jedoch nur solange kein Prozess abstürzt.

Bei der vorliegenden Implementierung sorgt ein Absturz eines Threads jedoch dafür, dass das Programm nicht mehr weiterlaufen kann (weil die Zählvariable nicht hochgesetzt wurde). Somit kann es zu einem Deadlock kommen und es ist somit möglich das ein (oder mehrere) Philosoph verhungert.

b.)

(1)

Bei Testalgorithmus 2 denken zuerst alle Philosophen. Wird ein Philosoph fertig nimmt er zuerst die linke Gabel und dann die rechte Gabel, außer es ist der mit Index 3, dann ist es anders herum. Schafft ein Philosoph dies, so isst er und gibt anschließend die Gabeln wieder frei. Ist eine Gabel die er nehmen möchte bereits in Benutzung, wartet er an dieser Stelle solange, bis diese Gabel wieder frei ist.

Hier nehmen also alle Philosophen zuerst die linke Gabel und dann die Rechte, bis auf den Thread mit Index 3, dort ist es andersherum. Dies hat zur Folge, dass es nicht mehr dadurch zu einem Deadlock kommen kann, da die Threads so durchschalten das jeder Philosoph die linke Gabel in der Hand hat und keiner mehr eine weitere in die Hand nehmen kann. Die Ausnahme bei Index 3 sorgt also dafür das mindestens ein Thread die zweite Gabel nehmen kann und es somit durch solch einen Fall nicht zu einem Deadlock kommt.

(2)

Auch hier kann ein Absturz eines Threads dazu führen, dass ein Philosoph verhungert, indem dadurch eine für einen anderen Philosophen benötigte Gabel nicht mehr freigegeben wird und dieser somit nicht mehr essen kann.

Somit ist auch dieser Algorithmus zwar nicht fair, aber trotzdem „fairer“ als der erste Algorithmus, da zumindest nicht das ganze Programm bei Absturz eines Threads nicht weiterlaufen kann, sondern nur die beiden jeweiligen „Sitznachbarn“ des abgestürzten „Philosophen“ betroffen sind.

Zudem können in dieser Implementierung mehrere Philosophen gleichzeitig essen (im Gegensatz zu Testalgorithmus 1) und es wird auch kein ineffizientes busy waiting verwendet.