|  |  |
| --- | --- |
| Labor Protokoll  Betriebssysteme Praktikum 2 Gruppe 2 Das Problem der Speisenden Philosophen | Labor Protokoll  Dies ist das Protokoll zum zweiten Laborversuch und umfasst eine kurze Beschreibung der Aufgabe, sowie eine Darstellung des Entwurfs und Diskussion des Ergebnisses  Steffen Giersch & Maria Lüdemann  HAW Hamburg |

Inhaltsverzeichnis

[Das Problem der Speisenden Philosophen 2](#_Toc372622711)

[Aufgabenstellung 2](#_Toc372622712)

[Der Entwurf 2](#_Toc372622713)

[*Darstellung des Entwurfs* 6](#_Toc372622714)

# Das Problem der Speisenden Philosophen

### Aufgabenstellung

Schreiben Sie ein Programm zur Darstellung des Problems der speisenden Philosophen

Das Problem der Speisenden Philosophen umfasst eine Anzahl Philosophen die rund um einen Tisch sitzen auf dem eine Schale Reis steht. Jeder Philosoph sitzt zwischen zwei Stäbchen jedoch gibt es genauso viele Stäbchen wie Philosophen, jeder von ihnen Denkt; irgendwann wird er hungrig und möchte beginnen zu essen. Dafür benötigt er allerdings zwei Stäbchen die er nur dann erhalten kann wenn weder sein rechter noch sein linker Nachbar gerade essen. Hat er es geschafft zu essen legt er seine Stäbchen ab und denkt wieder.

Die Philosophen werden hier durch Threads dargestellt die drei Zustände kennen Thinking, Hungry und Eat. Sie starten zu denken und wechseln dann nach einem festgelegten Intervall auf Hungry nun möchte der Thread die Ressourcen(Stäbchen) nehmen um zu essen. Dafür prüft er ob sie frei sind. Wenn ja nimmt er sie sich und sperrt sie für alle anderen. Wenn nicht dann verbleibt er im Hungry und wartet dort solange bis die beiden Threads neben ihm mit der Eat Schleife durch sind und ihn beim Stäbchen weglegen anstubsen. Dann nimmt er sie sich und beendet Hungry um zu Eat zu wechseln.

### Der Entwurf

*Array command[]*

*NPHILO 5*

*THINK\_LOOP 1000000000*

*EAT\_LOOP 500000000*

*CONSOLESIZEX 70*

*CONSOLESIZEY 40*

*ASCIICHARTOINTOFFSET 48*

*Array state[NPHILO]*

*Array command[NPHILO]*

*funktion main*

*lineCount = 0*

*NCurses.initialice*

*initialice(&mutex, &mutexPrint, &blockSem)*

*initialice(&barrierSingle, 2)*

*initialice(&barrierAll, NPHILO+1)*

*for 1..NPHILO*

*initialice(philo)*

*wait\_for(&barrierSingle)*

*endfor*

*wait\_for(&barrierAll)*

*while ch != 'q'*

*ch = getch*

*if ch == 'q'*

*for i = 0..NPHILO-1*

*command[i] = 'q'*

*endfor*

*else if ch == 'b'*

*i = getnum*

*command[i] = 'b'*

*else if ch == 'u'*

*i = getnum*

*command[i] = 'u'*

*sem\_post(&blockSem[i])*

*else if ch == 'p'*

*i = getnum*

*command[i] = 'p'*

*endif*

*endwhile*

*for i = 0..NPHILO-1*

*wait\_for\_thread(i)*

*endfor*

*destroy\_all\_synchronisers*

*NCruses.terminate*

*end main*

*funktion philo*

*state[philono] = 'T'*

*waitfor(&barrierSingle)*

*waitfor(&barrierAll)*

*while command[philono] != 'q'*

*for i..THINKLOOP*

*if command[philono] == 'b'*

*wait\_for(blockSem[philono])*

*else if command[philono] == 'p'*

*command[philono] = NULL*

*endForloop*

*endif*

*endfor*

*getSticks*

*while command[philono] != 'q'*

*for i..EATLOOP*

*if command[philono] == 'b'*

*wait\_for(blockSem[philono])*

*else if command[philono] == 'p'*

*command[philono] = NULL*

*endForloop*

*endif*

*endfor*

*endwhile*

*putsticks*

*endwhile*

*end philo*

*function changeAndDisplayStates(n, ch)*

*wait\_for(&mutexPrint)*

*state[n] = ch*

*lineCount += 1*

*if lineCount > CONSOLESIZEY*

*clearConsole*

*lineCount = 0*

*endif*

*print(states)*

*refreshConsole*

*for i = 0..NPHILO-1*

*if state[i] and state[i+1] == 'E'*

*exitWithError*

*endif*

*endfor*

*release(&mutexPrint)*

*end changeAndDisplayStates*

*function getSticks(no)*

*changeAndDisplayStates(no, 'H')*

*wait\_for(&mutex)*

*while leftStick and rightStick not 0*

*wait\_for(&condStick[no], &mutex)*

*endwhile*

*leftStick = 1*

*rightStick = 1*

*release(&mutex)*

*changeAndDisplayStates(no, 'E')*

*end getSticks*

*function putSticks(no)*

*wait\_for(&mutex)*

*changeAndDisplayStates(no, 't)*

*leftStick = 0*

*rightStick = 0*

*signal(leftPhilo)*

*signal(rightPhilo)*

*release(&mutex)*

*end putSticks*

### *Darstellung des Entwurfs*

Wie der Entwurf darstellt, versuchten wir uns erst an einer Lösung mit ncurses da die Ausgabe uns bedeutend besser gefiel, da ncurses auf eine neue Konsole ausgibt und nicht die Konsole vollschreibt, des Weiteren vermischt sich der ein- und ausgabe- Stream nicht und es ist nicht mehr notwendig mit einem Enter nach der Eingabe zu bestätigen.

Im elften Stock funktionierte diese Lösung auch einwandfrei doch stießen wir in den Praktikums Räumen leider auf einige Probleme. So hat der Rechner im siebten Probleme die ncurses Libary zu verwenden sodass es uns nicht möglich war das Programm neu zu compilieren. Die bereits compilierte Version funktionierte jedoch sofern man dem Rechner sehr viel Zeit zugestand da sie mit den großen Schleifen Zahlen sehr zu arbeiten hatten.

Somit schrieben wir die ncurses Version auf eine Lösung mit fgets um.