ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd

KIV/ZOS

Problém večeřících filozofů

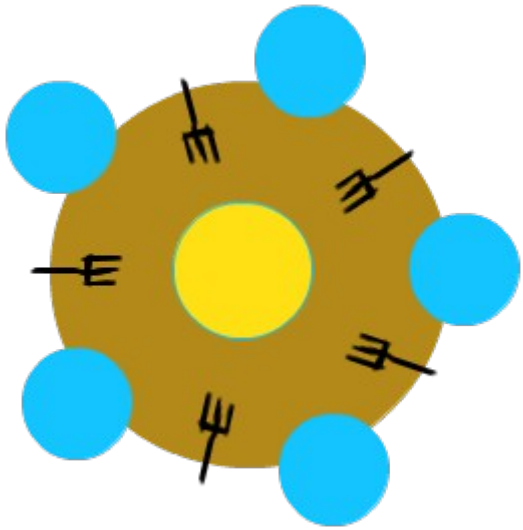
Autor: Antonín NEUMANN, A11B0439P
Akademický rok: 2013/2014

1 Zadání

Popis problému večeřících filozofů

Problém večeřících, rovněž se někdy uvádí obědvajících filozofů, byl definovaný v roce 1965 nizozemských informatikem Edsgerem Wybem Dijkstrou. Jedná se o jeden z klasických příkladů v informatice na ověření správné činnosti správy vláken.

V základní podobě je problém definován pro 5 filozofů, ti buď jedí nebo přemýšlejí, jídelní stůl s mísou špaget a 5 vidliček. Každý filozof má po své levé straně jednu vidličku a po pravé straně také jednu vidličku. Chce-li filozof jíst, musí zvednou svou levou i pravou vidličku.



Obrázek 1: Večeřící filozofové ilustrace

Požadavky na odevzdání

- program vytvoří nejméně 4 paralelní procesy nebo vlákna, ideálně s možností zadat při/po spuštění počet paralelních entit
- program bude průběžně informovat o činnosti jednotlivých vláken / procesů, tj. bude vidět, že všechna pracují korektně
- program bude zpracován tak, aby byl uživatel informován o tom, jaký problém program řeší a v čem jsou možná úskalí
- nutné ukázat kromě standardního běhu i možnost uvíznutí, aj.

2 Úvod

Vstupní parametry programu

Prvním vstupním parametrem programu je počet vláken (filozofů), který budou vytvořeny. Jako druhý parametr se vkládá počet jedení každého filozofa. Pokud již filozof jedl Nkrát, ukončí svojí činnost. Jedná se tedy o zastavovací podmínku, aby bylo možné zhodnotit výsledky. A posledním, tedy třetím, parametrem je typ simulace.

Ukázky spuštění programu:

- `./phil 5 2`
  Tímto příkazem se spustí standardní simulace s 5 vlákny/filozofy kdy každý filozof bude jíst dvakrát.
- `./phil 20 10 deadlock`
  Spuštění 20 vláken/filozofů, každý bude jíst 10krát, režim simulace je deadlock.
- `./phil help`
  Tímto příkazem program vypíše nápovědu ke svému ovládání.

Typy simulace

Program má v sobě implementovány celkem čtyři různé typy simulace.

1. Deadlock (parametr **deadlock**).
   Při zadání tohoto parametru, se program spustí v režimu, kdy záměrně dojde k situaci známé jako deadlock.
2. Starvation (parametr **starvation** nebo **vyhladoveni**)
   Při zadání tohoto parametru, se program spustí v režimu, kdy záměrně dojde k situaci známé jako vyhladovění (starvation).
3. Vyhrazeni částečného pořadí (parametr **order** nebo **poradi**)
   Po zadání tohoto parametru, se program spustí v režimu, kdy je pro správnou činnost vláken využito metody vyhrazení částečného pořadí.
4. Řešení za pomocí chráněné přístupu zdrojům (bez parametru)
   Pokud není zadán parametr typu simulace, provede program simulaci založenou na chráněném přístupu k vidličkám. Tento algoritmus je hlavní ukázkou řešení problému večeřících filozofů.

Deadlock

Tento stav nastává pokud se každý filozof pokusí zvednout například nejprve levou vidličku a poté pravou. Levou vidličku se podaří vzít každému filozofovi, ale pravou již žádnému z nich. Všichni se tedy dostanou do mrtvého bodu, kdy nemůžou ani jíst ani přemýšlet.

Starvation

Vyhladovění je stav, který nastává, když se vlákno ani po určité době nemůže dostat ke zdrojům.

V tomto případě nedochází sice k deadlocku, protože filozof jde při nezískání vidliček opět přemýšlet, ale může dojít k vyhladovění, jelikož se ani jeden z filozofů opakovaně nedokáže najíst.

Vyhrazení částečného pořadí

Každý filozof si vždy bere nejdříve vidličku s nižším číslem a potom až vidličku s vyšším číslem. Pokládání vidliček probíhá v opačném pořadí.

Každý filozof se tedy nejprve pokusí zvednou levou vidličku, až na posledního, který se pokusí zvednou jako první pravou vidličku. Tím je zaručeno, že alespoň jednomu filozofovi se podaří zvednout obě vidličky a najíst se. Filozof, který se najedl, pouští vidličky v opačném pořadí než je zvedl, toto chování umožní najíst se jeho kolegovi a postupně i všem dalším filozofům.

Chráněný přístup ke zdrojům

Tento algoritmus se někdy též označuje jako „číšník“, kdy se použije jeden další semafor (číšník) pro přístupu ke zdrojům (vidličkám). Tento algoritmus jsem si vybral jako hlavní řešení Problému večeřících filozofů v této samostatné práci.

Přidáním dalšího semaforu docílíme následujícího chování:

Filozof si vyžádá od číšníka přístup k vidličkám, pokud zatím nikdo jiný vidličky nezvedá je mu umožněn přístup, v opačném případě jde filozof přemýšlet. Následně se filozof pokusí zvednou levou vidličku, když se mu to nepodaří jde opět přemýšlet, jinak se pokusí zvednout vidličku pravou. Je-li mu pravá vidlička nepřístupná, položí levou a jde přemýšlet, jinak jde jíst a uvolní číšníka.

Při použití dalšího semaforu se vyhneme nežádoucím stavům jako je uvíznutí (deadlock) a vyhladovění (starvation). Zvedání vidliček je v tomto případě vlastně jakousi atomickou operací.

3 Závěr

Tato semestrální práce mě naučila, jak se v programovacím jazyce C pracuje s vlákny. Jak se vlákna vytváření, jak se s nimi pracuje a jak správně synchronizovat přístup vláken ke společným zdrojům. Přístup ke společným zdrojům jsem řešil za pomoci semaforů, konkrétně bitových semaforů z knihovny semaphore.h.