Systemy Rozproszone

Prowadzący
Dr. Radosław Kycia

RAPORT

Problem ucztujących filozofów przy użyciu procesów UNIX/POSIX

Autorzy:
Igor Boradyn 155971

Jerzy Dębowski 151266

Wstęp

Problem ucztujących filozofów jest klasycznym przykładem zadania snchronizacji procesów. Problem synchronizacji procesów pojawia się wszędzie tam, gdzie mamy do czynienia ze współpracującymi ze sobą współbieżnymi procesami. Najczęściej spotykane przyczyny, dla których konieczna jest synchronizacja współpracujących procesów to np. Sytuacje w których procesy współdzielą pewną strukturę danych, wyniki działania jednego procesu stanowią dane dla innego procesu lub procesy korzystają z pewnej wspólnej puli zasobów, które pobierają i zwalniają wedle potrzeb.

Problem pięciu filozofów, to akademicki problem synchronizacji. Mamy pięciu filozofów. Każdy z nich rozmyśla. Gdy zgłodnieje, to idzie jeść, a gdy się naje, to kontynuuje rozmyślania, aż znowu nie zgłodnieje itd. Filozofowie jedzą przy okrągłym stole, przy czym każdy z nich ma swoje miejsce. Każdy z nich najpierw nakłada sobie jedzenie z miski stojącej na środku stołu, bierze dwa widelce leżące po lewej i prawej stronie talerza i je. Po zjedzeniu odkłada widelce na miejsce. Rzecz w tym, że na stole jest tylko pięć pałeczek, po jednej między dwoma sąsiednimi talerzami. Problem polega na takim zsynchronizowaniu poczynań filozofów, aby wykluczyć możliwość blokady i zagłodzenia.

Program

Program został napisany w języku C. Do zarządzania dostępem do widelców I zapobiegania zakleszczeniom użyto mutex-ów. Zaimplementowane są w nim biblioteki **`stdio.h`**, **`stdlib.h`**, **`unistd.h`** oraz najważniejsza **`pthread.h`**. Na wstępie definiowane są stałe **`N`** (liczba filozofów oraz widelców), **`EATING_TIME`** (czas jedzenia), **`THINKING_TIME`** (czas rozmyślania) oraz tablica mutex-ów **`forks`**. Każdy mutex reprezentuje widelec, który może być trzymany tylko przez jednego filozofa.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

#define N 5
#define EATING_TIME 3
#define THINKING_TIME 2

pthread_mutex_t forks[N];

```

Funkcja philosopher jest funkcją, która będzie wykonywana przez każdego filozofa jako osobny wątek. Na początku funkcji pobierany jest identyfikator filozofa z argumentu, a następnie przypisywane są mu odpowiednie widelce. Następnie w nieskończone pętli filozof czeka na widelce i blokuje je poprzez **`pthread_mutex_lock`**. Jeśli widelce są dostępne, filozof zaczyna jeść przez określony czas, a następnie odkłada widelce jednocześnie odblkowując je poprzez **`pthread_mutex_unlock`** i zaczyna rozmyślać przez wcześniej ustalony czas.

```c
void *philosopher(void *arg) {
    int id = *((int *)arg);
    int right_fork = id;
    int left_fork = (id + 1) % N;

    while (1) {
        printf("Filozof %d czeka na widelce %d i %d.\n", id, right_fork, left_fork);
        pthread_mutex_lock(&forks[right_fork]);
        pthread_mutex_lock(&forks[left_fork]);

        printf("Filozof %d zaczyna jeść.\n", id);
        sleep(EATING_TIME);

        printf("Filozof %d odkłada widelce %d i %d.\n", id, right_fork, left_fork);
        pthread_mutex_unlock(&forks[right_fork]);
        pthread_mutex_unlock(&forks[left_fork]);

        printf("Filozof %d zaczyna myśleć.\n", id);
        sleep(THINKING_TIME);
    }
}

```

W funkcji main na początku inicjowane są odpowiednie mutex-y dla wszystkich widelców poprzez funkcję **`pthread_mutex_init.`** Następnie przy pomocy **`pthread_create`** tworzone są odpowiednie wątki dla każdego filozofa, a na końcu została użyta funkcja **`pthread_join`**, aby czekać na zakończenie wszystkich wątków filozofów.

```c
int main() {
    pthread_t philosophers[N];

    int ids[N];
    int i;

    for (i = 0; i < N; i++)
        pthread_mutex_init(&forks[i], NULL);

    for (i = 0; i < N; i++) {
        ids[i] = i;
        pthread_create(&philosophers[i], NULL, philosopher, (void *)&ids[i]);
    }

    for (i = 0; i < N; i++)
        pthread_join(philosophers[i], NULL);

    return 0;
}
```

Przykładowy wynik programu

```
jerzy@jerzy-pc:~/Desktop$ ./filozof
Filozof 2 czeka na widelce 2 i 3.
Filozof 2 zaczyna jeść.
Filozof 1 czeka na widelce 1 i 2.
Filozof 0 czeka na widelce 0 i 1.
Filozof 3 czeka na widelce 3 i 4.
Filozof 4 czeka na widelce 4 i 0.
Filozof 2 odkłada widelce 2 i 3.
Filozof 2 zaczyna myśleć.
Filozof 1 zaczyna jeść.
Filozof 2 czeka na widelce 2 i 3.
Filozof 1 odkłada widelce 1 i 2.
Filozof 1 zaczyna myśleć.
Filozof 0 zaczyna jeść.
Filozof 1 czeka na widelce 1 i 2.
Filozof 0 odkłada widelce 0 i 1.
Filozof 0 zaczyna myśleć.
Filozof 4 zaczyna jeść.
Filozof 0 czeka na widelce 0 i 1.
Filozof 4 odkłada widelce 4 i 0.
Filozof 4 zaczyna myśleć.
Filozof 3 zaczyna jeść.
Filozof 4 czeka na widelce 4 i 0.
^C
```

Źródła:

https://pl.wikipedia.org/wiki/Problem_ucztuj%C4%85cych_filozof%C3%B3w

https://edu.pjwstk.edu.pl/wyklady/sop/scb/wyklad5/index.html

https://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread.h.html