PCS 3868 – Sistemas de Computação de Alto Desempenho

Nome: Luciana Marques e Guillaume Hillion
Aula pratica : CPAR & Paralelismo

Exercico 2 - Codigo

```
#include <stdio.h>
#include <semaphore.h>
#include <stdlib.h>
#include <omp.h>
#define think 20
#define eat 10

sem_t sem[5];

void thinking(i)
{
	printf("filosofo %d thinking\n",i);
	fflush(stdout);
	usleep(think);
}

void eating(int i)
{
	printf("filosofo %d eating\n",i);
	fflush(stdout);
	usleep(eat);
}

void filosofo(int i)
{
	int k;

	for (k=0;k<100;k++) {
		thinking(i);
		if (i != 0){
			sem_wait(&sem[i]);
			usleep(1);
			sem_wait(&sem[(i+1)%5]);
		}
		else
		{
			sem_wait(&sem[(i+1)%5]);
			usleep(1);
			sem_wait(&sem[i]);
```

```
                }
                eating(i);
                sem_post(&sem[i]);
                sem_post(&sem[(i+1)%5]);
        }
}

void main()
{

        printf("INICIO \n");

        fflush(stdout);

        sem_init(&sem[0],0,1);
        sem_init(&sem[1],0,1);
        sem_init(&sem[2],0,1);
        sem_init(&sem[3],0,1);
        sem_init(&sem[4],0,1);


        #pragma omp parallel num_threads(5)

        {
                #pragma omp sections
                {
                 #pragma omp section
                 filosofo(0);
                                #pragma omp section
                                filosofo(1);
                 #pragma omp section
                 filosofo(2);
                        #pragma omp section
         filosofo(3);
                        #pragma omp section
         filosofo(4);
    }
  }
     printf("FIM\n");
}
```

Exercico 2 - Execucao

O Exercício 2 visa colocar em prática uma solução para o problema lógico dos filósofos.

Considere 5 filósofos que passam a vida a pensar e a comer. Partilham uma mesa redonda rodeada por 5 cadeiras sendo que cada uma das cadeiras pertence a um filósofo. No centro da mesa encontra-se uma taça de arroz e estão 5 garfos na mesa, um para cada filósofo.

Quando um filósofo pensa não interage com os seus colegas. De tempos em tempos, cada filósofo fica com fome e tenta apanhar os dois garfos que estão mais próximos (os garfos que estão ou à esquerda ou à direita). O filósofo apenas pode apanhar um garfo de cada vez e como o leitor compreende, não pode apanhar um garfo se este estiver na mão do vizinho. Quando um filósofo esfomeado tiver 2 garfos ao mesmo tempo ele come sem largar os garfos. Apenas quando acaba de comer, o filósofo pousa os garfos, libertando-os e começa a pensar de novo. O nosso objetivo é ter uma representação/implementação que nos permita simular este jantar sem que haja problemas de deadlock ou starvation.

A execução inicial do código resulta em um deadlock.

INICIO
filosofo 0 thinking
filosofo 1 thinking
filosofo 2 thinking
filosofo 3 thinking
filosofo 4 thinking

Nós implementamos uma solução simples: a solução assimétrica.

Essa é uma das soluções mais simples desse problema.

Solução Assimétrica

- Os 4 primeiros filósofos → idem a primeira solução (primeiro toma fork a esquerda e depois a direita);
- o quinto filósofo é diferente (primeiro toma fork a direita e depois a esquerda)

Essa solução elimina o risco de deadlocks.

Aqui está o resultado com 10 iterações por filósofo.

INICIO
filosofo 0 thinking
filosofo 1 thinking
filosofo 2 thinking
filosofo 3 thinking
filosofo 0 eating
filosofo 3 eating
filosofo 0 thinking
filosofo 3 thinking
filosofo 2 eating
filosofo 2 thinking
filosofo 1 eating
filosofo 1 thinking
filosofo 3 eating
filosofo 0 eating

filosofo 3 thinking
filosofo 2 eating
filosofo 0 thinking
filosofo 2 thinking
filosofo 1 eating
filosofo 1 thinking

filosofo 3 eating
filosofo 0 eating
filosofo 3 thinking
filosofo 2 eating
filosofo 0 thinking
filosofo 1 eating
filosofo 2 thinking
filosofo 1 thinking
filosofo 3 eating
filosofo 0 eating
filosofo 3 thinking
filosofo 2 eating
filosofo 0 thinking
filosofo 1 eating
filosofo 2 thinking
filosofo 3 eating
filosofo 1 thinking
filosofo 0 eating
filosofo 4 thinking
filosofo 2 eating
filosofo 1 eating
filosofo 4 eating
filosofo 4 thinking
filosofo 4 eating
filosofo 4 thinking
filosofo 4 eating
filosofo 4 thinking
filosofo 4 eating
filosofo 4 thinking
filosofo 4 eating
FIM