Entrega 2
Programação Concorrente e Distribuída

Matheus Felipe Rodrigues 93943

Atividade 01:

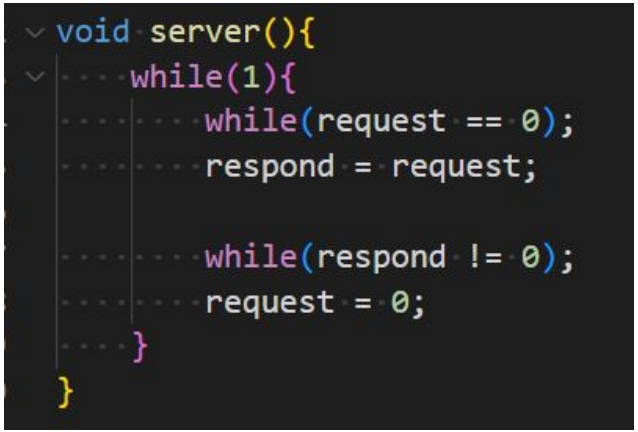
Seção Crítica por Espera Ocupada

Atividade 01:
Variáveis Globais e Funções

```c
#define NUMTHREADS 4

// globals
int soma = 0;
int request = 0;
int respond = 0;
```

```c
// headers
void server();          // server process
void critical(int id);  // critical section
void client(int id);    // client process
```

Atividade 01:
Função Server



```
void server(){
    while(1){
        while(request == 0);
        respond = request;

        while(respond != 0);
        request = 0;
    }
}
```

Atividade 01:
Função Client

```c
void client(int id){
    while(1){
        // pré protocolo
        while(respond != id) request = id; // comentar esta linha para remover o controle da secao critica!!

        // secao critica
        critical(id);
        printf("ID: %d / SOMA: %d\n", id, soma);

        // pos protocolo
        respond = 0;
    }
}
```

Atividade 01:
Função Critical

```
void critical(int id){
    int local = soma;
    sleep(rand()%2);
    soma = local + 1;
}
```

Atividade 01:
Função Main

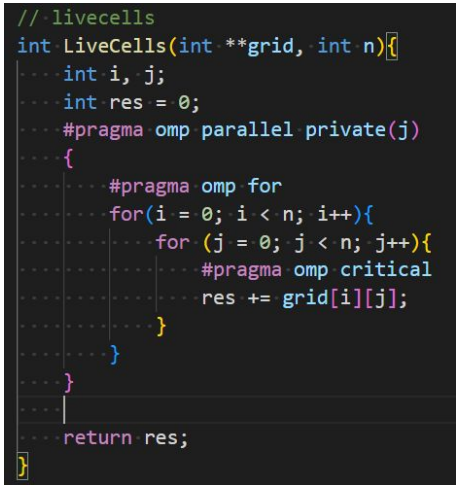
```c
int main(){
    omp_set_num_threads(NUMTHREADS);

    #pragma omp parallel
    {
        int id = omp_get_thread_num();

        if(id == 0) server();
        else client(id);
    }

    return 0;
}
```

Atividade 01:
Demonstração

Atividade 02:

Somatórias, Seção Crítica e Reduções em OpenMP

Atividade 02:
#pragma omp critical



```
// livecells
int LiveCells(int **grid, int n){
    int i, j;
    int res = 0;
    #pragma omp parallel private(j)
    {
        #pragma omp for
        for(i = 0; i < n; i++){
            for (j = 0; j < n; j++){
                #pragma omp critical
                res += grid[i][j];
            }
        }
    }

    return res;
}
```

Atividade 02:
#pragma omp for reduction

```c
// livecells
int LiveCells(int **grid, int n){
    int i, j;
    int res = 0;
    #pragma omp parallel private(j) reduction(+:res)
    {
        #pragma omp for
        for(i = 0; i < n; i++){
            for (j = 0; j < n; j++){
                res += grid[i][j];
            }
        }
    }
    return res;
}
```

Atividade 02:
Tabela de Dados

			THREADS			
			1	2	4	8
game of life	critical	Tempo (s)	278	361	549	554
		Speedup (s)	1	0.77	0.506	0.501
	reduction	Tempo (s)	131	66	44	38
		Speedup (s)	1	1.985	2.977	3.447

Atividade 03:
Seção Crítica em Java

Atividade 03:
Variáveis e Funções

```
public class TrafficController {
    final Lock lock = new ReentrantLock();
    final Condition condition = lock.newCondition();

    int state = 0; // 0 - vazia, 1 - cheia
    ...
    public void enterLeft();
    public void enterRight();
    public void leaveLeft();
    public void leaveRight();
```

Atividade 03:
Lado Esquerdo

```
public void enterLeft(){
    lock.lock();

    try {
        if ((state != 0)) condition.await();
    } catch (InterruptedException ie) {
        System.err.println(ie.toString());
    }

    state = 1;
    lock.unlock();
}
```

```
public void leaveLeft(){
    lock.lock();
    state = 0;
    condition.signal();
    lock.unlock();
}
```

Atividade 03:
Lado Direito

```
public void enterRight(){
    lock.lock();

    try {ie
        if ((state != 0)) condition.await();
    } catch (InterruptedException ie) {
        System.err.println(ie.toString());
    }

    state = 1;
    lock.unlock();
}
```

```
public void leaveRight(){
    lock.lock();
    state = 0;
    condition.signal();
    lock.unlock();
}
```

Atividade 03:
Demonstração