LICENCIATURA EM CIÊNCIAS DA COMPUTAÇÃO

Programação Concorrente

Teste de Avaliação[1]

17 de Maio de 2022 Duração: 2h00m

I

1 Explique o conceito de *deadlock* e descreva um cenário onde este fenómeno poderá acontecer.

2 Compare brevemente monitores modernos (que aparecem em linguagens populares actuais) com variantes clássicas. Explique o que complica a implementação de uma barreira reutilizável usando monitores modernos.

3 Descreva a utilidade do *selective receive* suportado por Erlang.

II

Pretende-se que escreva em Java, fazendo uso de primitivas baseadas em monitores, código que permita o agrupamento, aos pares, de threads que têm o papel de produtor ou consumidor sobre um *bounded-buffer* criado para cada par de threads. Para tal, implemente a seguinte interface:

```
interface MatchMaker {
    BoundedBuffer waitForConsumer();
    BoundedBuffer waitForProducer();
}
```

A operação `waitForConsumer()`, para ser utilizada por uma thread com o papel de produtor, deverá bloquear até este produtor poder ser emparelhado com um consumidor (uma thread que tenha invocado ou vá invocar o `waitForProducer`), devolvendo um *BoundedBuffer*, que deverá ser criado para uso exclusivo deste par de threads. De igual modo, `waitForProducer`, para ser usada por um consumidor, deverá bloquear até este poder ser emparelhado com um produtor. Assuma um número arbitrário de threads que possa querer ser emparelhada, faça o emparelhamento por ordem de chegada, e evite acordar threads sem necessidade. Assuma a existência de uma classe `BoundedBuffer` com um construtor por omissão.

III

Apresente o código Erlang para implementar o mesmo cenário descrito no grupo II, através de um ou mais processos. Suponha que os clientes são processos Erlang que comunicam pelo mecanismo nativo de mensagens, e implemente também as funções de interface apropriadas para serem usadas por estes para interagirem com o(s) processos relevante(s).

---

[1] Cotação — 6+8+6