Trabalho II de Computação Concorrente:

Gabriel da Fonseca Ottoboni Pinho - DRE 119043838
Rodrigo Delpreti de Siqueira - DRE 119022353

21/05/2021

1 Leitores e escritores

1.1 O problema

O modelo de leitores e escritores tem como objetivo coordenar o acesso a um certo recurso entre várias threads. No caso desse trabalho, o recurso é o array que contém as medições dos sensores.

A necessidade de coordenar os acessos a esse array existe por conta de condições de corrida que podem ocorrer quando uma thread tenta ler/escrever enquanto outra thread já estava escrevendo no array. Dessa forma, temos os seguintes requisitos:

- Se alguma thread estiver escrevedo, nenhuma outra thread pode ler ou escrever.
- Se uma ou mais threads estiverem lendo, nenhuma outra thread pode escrever.

Sabendo disso, é necessário um sistema no qual cada thread pede permissão para acessar o array e, caso o acesso não seja possível naquele momento, a thread é bloqueada.

1.2 A solução

A solução implementada no trabalho consiste em um *struct* `Rw` que armazena os números de threads atualmente lendo e escrevendo. Esses números são atualizados por meio de funções que as threads chamam antes e depois de iniciar suas tarefas.

Internamente, o controle de acesso é feito utilizando os *structs* `pthread_mutex_t` e `pthread_cond_t`. O primeiro garante exclusão mútua durante o acesso às variáveis que armazenam os números de leitores e escritores, enquanto que o segundo é utilizado para controlar o bloqueio das threads que aguardam permissão para ler/escrever.

```
Rw rw;
rw_init(&rw);

rw_get_read(&rw);
/* Ler à vontade */
rw_release_read(&rw);

rw_get_write(&rw);
/* Escrever à vontade */
rw_release_write(&rw);

rw_destroy(&rw);
```

Como mostrado no exemplo acima, a função `rw_init` inicializa os campos do *struct*, enquanto que `rw_destroy` faz o oposto. Antes de ler, a thread deve chamar `rw_get_read`, que retornará imediatamente se nenhuma outra estiver escrevendo. Caso contrário, a thread ficará bloqueada até que nenhuma outra thread esteja escrevendo. Ao acabar de ler, a thread deve chamar `rw_release_read`. As funções `rw_get_write` e `rw_release_write` funcionam de forma análoga.

Quando há alguma thread aguardando permissão para escrever, a prioridade para escrita é garantida de duas formas:

- Assim que o último leitor chama `rw_release_read`, é garantido que uma thread será liberada para escrita.
- Assim que um escritor chama `rw_release_write`, é garantido que uma thread será liberada para escrita.

1.3 Testes

Os testes realizados sobre a implementação que criamos para o padrão de leitores/escritores visam atender os seguintes requisitos:

1. mais de um leitor pode ler ao mesmo tempo uma área de dados compartilhada;
2. apenas um escritor pode escrever de cada vez nessa mesma área;
3. enquanto o escritor está escrevendo, os leitores não podem ler.

Estes requisitos são avaliados individualmente, no arquivo separado *rwtest.c*.

2 Monitoramento de temperatura

A função `check_temperature` verifica se haverá um caso de sinal vermelho, amarelo ou condição normal, com base nos requisitos do trabalho. O estado exibido no console apenas é atualizado quando ocorre uma mudança.

3 Discussão

Durante a execução, observamos que o Alerta Vermelho é raramente acionado.

4 Referências Bibliográficas