Programação Concorrente – Aula 10

Gilmar Gimenes Rodrigues

1 Problema dos Leitores Escritores

- Modela o acesso concorrente a um banco de dados.
- Varias threads.
- Leitores so leem dados do banco de dados.
- Escritores podem ler e alterar dados do banco de dados.

Deseja-se que:

- Somente um escritor tenha acesso ao banco de dados ao mesmo tempo.
- Um leitor e um escritor não acessem o banco de dados ao mesmo tempo.
- Vários leitores podem acessar o banco de dados ao mesmo tempo.

```
(n_leitores == 0 || n_escritores == 0) && n_escritores <= 1)

leitor() {
    for(;;) {
        protocolo_de_entrada_no_BD_p_leitura();
        acessa_BD_p_leitura();
        protocolo_de_saida_do_BD_p_leitura();
    }
}

escritor() {
    for(;;) {
        protocolo_de_entrada_no_BD_p_escrita();  // P(acesso_ao_BD)
        acessa_BD_p_escrita();
        protocolo_de_saida_do_BD_p_escrita();    // V(acesso_ao_BD)
    }
}

Tentativa:

semaforo acesso_ao_BD = 1;
int n_leitores = 0;

leitor() {
    for(;;) {
        if(n_leitores_ao_BD);
```

```
            P(acesso_ao_BD);
        n_leitores++;
        acesso_BD_p_leitura();
        n_leitores--;
        if(n_leitores == 0) {
            V(acesso_ao_BD);
}
```

Solução:

```
semaforo acesso_ao_BD = 1;
semafor mutex = 1;

leitor() {
    for(;;) {
        P(mutex);
        if(leitores == 0)
             P(acesso_ao_BD);
        n_leitores++;
        V(mutex);
        acessa_BD_leitura();
        P(mutex);
        n_leitores--;
        if(n_leitores == 0)
             V(acesso_ao_BD);
        V(mutex);
    }
}

escritor() {
    for(;;) {
        P(acessa_ao_BD);
        acessa_BD_p_escrita();
        V(acesso_ao_BD);
    }
}
```

2 Problemas dos filósofos (Dinning Philosophers)

Modela o acesso concorrente a conjuntos de variáveis com intersecção não nula.

```
filosofo(int i) {
    for(;;) {
        pegar_garfos();
        comer();
        soltar_garfos();
        pensar();
    }
}
```

```
#define N 5;

semaforo garfo[N] = {1,1,...,1}

filosofo(int i) {
   for(;;) {
       P(garfo[i]);
       P(garfo[i-1]);  // Aqui se i == 0 trocar i-1 por N-1.
       come();
       V(garfo[i]);
       V(garfo[i-1]);  // mesma coisa do de cima.
       pensar();
    }
}

filosofo_canhoto() {
    for(;;) {
       P(garfo[3]);
       P(garfo[4]);
       come();
       V(garfo[3]);
       V(garfo[4]);
    }
}
```

Retomando Leitores/Escritores:

```
int n_leitores = 0;
int n_escritores = 0;

leitor() {
   for(;;) {
       <await n_escritores == 0; n_leitores++;>;
       acessa_BD_p_leitura();
       <n_leitores--;>
   }
}

escritor() {
    for(;;) {
        <await(n_leitores == 0 && n_escritores == 0); n_escritores++;>
        acesso_ao_BD_p_escrita();
        <n_escritores-->
    }
}
```

```
int n_leitores = 0;
int n_escritores = 0;
```

```
semaforo mutex = 1;
semaforo leitura = 0;  // para n_esritores == 0
semaforo escrita = 0;  // para (n_leitores == 0 && n_escritores == 0)

int leitores_esperando = 0;
int escritores_esperando = 0;

leitor() {
    for(;;) {
        P(mutex);
        if(n_escritores > 0) {
            leitores_esperando++;
            V(mutex);
            P(leitura);
        }
        n_leitores++;
        if(leitores_esperando > 0) {
            leitores_esperando--;
            V(leitura)
        }
        else
            V(mutex);
        acessa_BD_para_leitura();
        P(mutex);
        n_leitores--;
        if(n_leitores == 0 && escitores_esperando > 0) {
            escritores_esperando--;
            V(escrita);
        }
        else
            V(mutex);
    }
}
```