**Relatório Técnico: Problema Produtor-Consumidor**

**1. Introdução / Motivação**

**1.1 Descrever do que se trata**

O problema Produtor-Consumidor envolve a coordenação de processos que produzem dados (produtores) e processos que consomem esses dados (consumidores). A principal dificuldade reside em garantir que os produtores e consumidores trabalhem, sem que os produtores sobrecarreguem os consumidores ou que os consumidores tentem consumir dados inexistentes.

**1.2 Apresentar os principais desafios**

1. **Sincronização**: Assegurar que os produtores e consumidores acessem a fila compartilhada de maneira segura, evitando condições de corrida.
2. **Gerenciamento de Tempo**: Controlar os tempos de produção e consumo para manter um fluxo eficiente de dados.
3. **Equilíbrio de Carga**: Garantir que a fila de produção não seja sobrecarregada e que os consumidores não fiquem ociosos sem dados para consumir.

## 2. Descrição e apresentação dos algoritmos

### 2.1 Contextualizar a solução para os desafios

A solução proposta, utiliza o Erlang, por suas capacidades de programação concorrente e distribuída. Erlang facilita a criação de processos e a comunicação entre eles, oferecendo mecanismos eficientes para a sincronização.

### Algoritmo do Produtor

Os produtores entram em um loop onde produzem itens periodicamente e os adicionam a uma fila. Quando um consumidor solicita um item e a fila não está vazia, o item é enviado ao consumidor.

### Algoritmo do Consumidor

Os consumidores entram em um loop onde solicitam itens aleatoriamente de um dos produtores. Após receber um item, o consumidor consome o produto, respeitando o tempo de consumo determinado pelo tipo do produto.

## 3. Conclusão / Relato de experiência com a linguagem

### 3.1 Experiência com a linguagem

Erlang se mostrou uma excelente escolha para resolver o problema do Produtor-Consumidor devido à sua robustez e simplicidade na criação e gerenciamento de processos concorrentes. A linguagem facilita a comunicação entre processos e oferece mecanismos eficientes para sincronização.

### 3.2 Um exemplo de aplicação dos problemas estudados

Um exemplo prático de aplicação deste problema é em sistemas de processamento de pedidos em comércio eletrônico, onde múltiplos servidores (produtores) processam pedidos e múltiplos serviços (consumidores) tratam diferentes etapas do processamento, como validação de pagamento, verificação de estoque e embalagem.

## 4. Código Fonte

### explicações de como executar o projeto/fonte

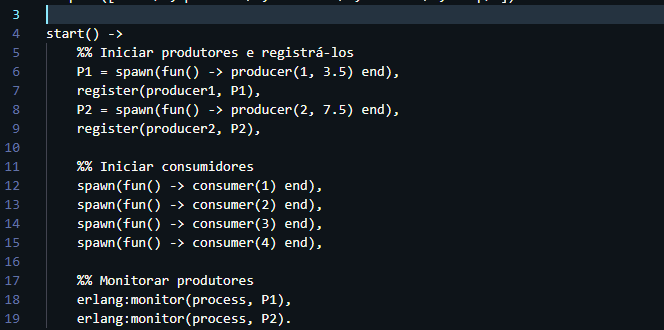
O código Erlang fornecido implementa o problema Produtor-Consumidor, onde dois produtores geram itens de diferentes tipos e quatro consumidores consomem esses itens. Vamos analisar cada parte do código detalhadamente.

Módulo e Exportação



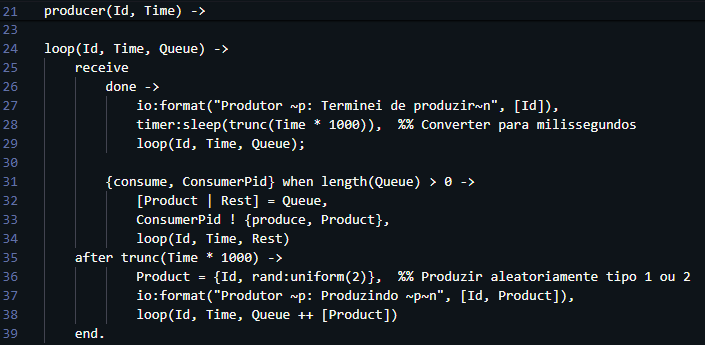
* **-module(prodcons)**: Define o nome do módulo como prodcons.
* **-export([...])**: Lista as funções que serão exportadas, tornando-as acessíveis fora do módulo.

Função start/0



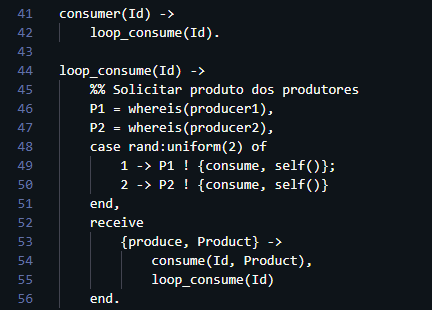
* **spawn(fun() -> producer(...))**: Cria novos processos para os produtores, que serão executados em paralelo.
* **register(producerX, P1/P2)**: Registra os processos dos produtores com nomes (producer1 e producer2) para facilitar a comunicação.
* **spawn(fun() -> consumer(...))**: Cria novos processos para os consumidores.
* **erlang (process, P1/P2)**: Monitora os processos dos produtores para detectar falhas.

Função producer/2 e loop/3



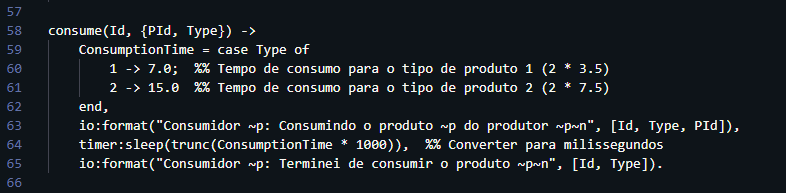
* **producer(Id, Time)**: Inicia o loop do produtor com o identificador e o tempo de produção.
* **loop(Id, Time, Queue)**: Loop principal que gerencia a produção de itens e responde a pedidos de consumidores.
* **receive ... end**: Aguarda mensagens de outros processos.
* **done**: Mensagem para finalizar a produção.
* **{consume, ConsumerPid}**: Mensagem de um consumidor solicitando um item.
* **after ...**: Produz um novo item após um determinado tempo (Time), adicionando-o à fila (Queue).

Função consumer/1 e loop\_consume/1



* **consumer(Id)**: Inicia o loop do consumidor com o identificador.
* **loop\_consume(Id)**: Loop principal que gerencia a solicitação e consumo de itens.
* **whereis(producerX)**: Obtém o Pid dos produtores registrados.
* **case rand**
* **(2)**: Solicita um item aleatoriamente de um dos produtores.
* **receive ... end**: Aguarda a mensagem de um produtor contendo o item solicitado.

Função consume/2



**consume(Id, {PId, Type})**: Consome o item recebido, calculando o tempo de consumo com base no tipo de produto.

* **ConsumptionTime**: Define o tempo de consumo com base no tipo do produto.
* **Io (...)**: Exibe mensagens de status no console.
* **Timer (...)**: Simula o tempo de consumo.

Como Executar o Projeto/Fonte

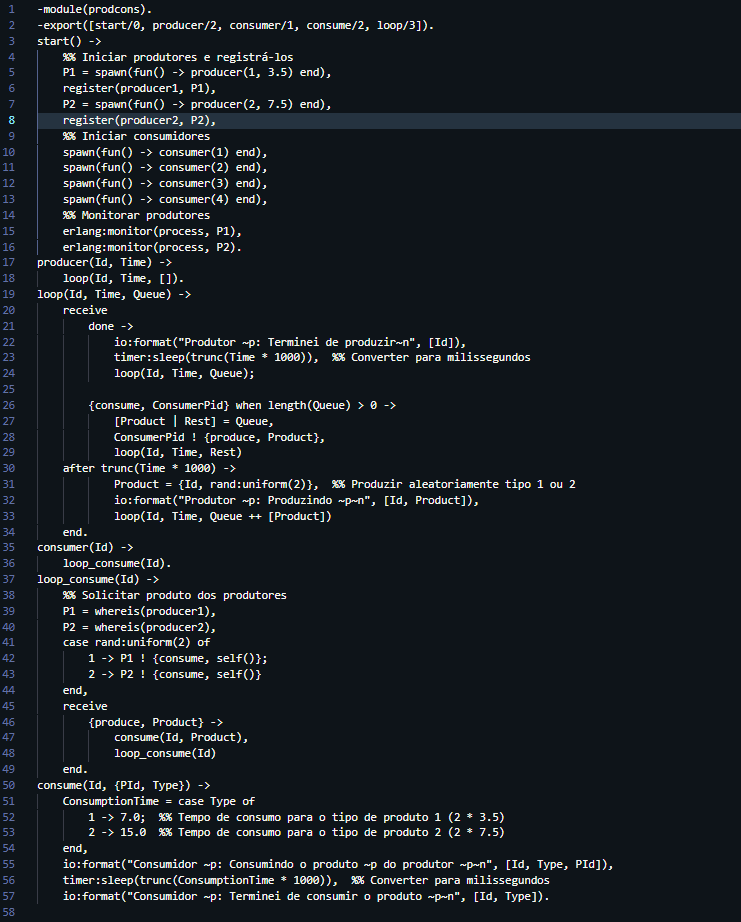
* Navegue até o diretório onde o arquivo prodcons.erl está salvo.
* Inicie o shell do Erlang digitando erl no terminal.
* Compile o módulo com o comando:



Execute a função start/0 para iniciar os processos dos produtores e consumidores:



Código Fonte



Resultado

