Ferramenta educacional para auxiliar no ensino do modelo fork-join

Lucas da Silva Marcos[1], **Rodolfo Adamshuk Silva**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Dois Vizinhos, Paraná

lucasmarcos@alunos.utfpr.edu.br, rodolfoa@utfpr.edu.br

Abstract.

Resumo. *A importância da programação concorrente no campo da ciência da computação é inegável. No entanto, a complexidade inerente ao ensino desta disciplina em ambientes acadêmicos é um desafio significativo. Este artigo propõe um compilador como ferramenta educacional destinada a facilitar o ensino do modelo de programação fork-join, um paradigma utilizado na programação concorrente.*

1. Introdução

2. Trabalhos relacionados

[Van Steen and Tanenbaum 2017].

fork-join [Conway 1963] e [Dennis and Van Horn 1966].

Compiladores [Louden 2004].

Recursos educacionais para ensino de programação.

Trabalhos relacionados.

Perguntas:

Qual é o referencial teorico?

Quais são os paradigmas de programação concorrente?

Os principais paradigmas de programação concorrente incluem:

Memória Compartilhada: Neste paradigma, os processos concorrentes compartilham um espaço de memória comum, permitindo a comunicação direta entre eles por meio da leitura e escrita em variáveis compartilhadas.

Passagem de Mensagem: Neste paradigma, os processos concorrentes se comunicam trocando mensagens entre si, sem compartilhar diretamente memória. As mensagens podem ser enviadas e recebidas através de canais de comunicação específicos.

Qual a diferença entre memória compartilhada e passagem de mensagen?

A principal diferença entre memória compartilhada e passagem de mensagem reside na forma como os processos concorrentes se comunicam:

Memória Compartilhada: Os processos acessam e modificam variáveis armazenadas em um espaço de memória compartilhado. A comunicação ocorre diretamente por meio da leitura e escrita nessas variáveis.

Passagem de Mensagem: Os processos se comunicam trocando mensagens entre si, enviando dados de um processo para outro através de canais de comunicação. Não há compartilhamento direto de memória entre os processos.

QUal a diferença entre problema, tarefa e processo?

Problema: Refere-se a uma questão ou desafio a ser resolvido por um sistema computacional.

Tarefa: É uma unidade de trabalho dentro de um programa ou sistema, geralmente relacionada à resolução de uma parte específica de um problema maior.

Processo: É uma instância em execução de um programa em um sistema computacional. Cada processo tem seu próprio espaço de memória e fluxo de execução, podendo realizar múltiplas tarefas de forma concorrente.

Um exemplo de código fork-join.

Quais são as fases de um compilador?

As fases de um compilador incluem:

Análise Léxica Análise Sintática Análise Semântica Geração de Código Intermediário Otimização de Código Geração de Código Executável

O que é a analise léxica?

A análise léxica é a primeira fase do processo de compilação, onde o código-fonte é lido e convertido em tokens. Esses tokens representam os componentes básicos da linguagem, como palavras-chave, identificadores, operadores e símbolos especiais.

O que é a analise sintática?

A análise sintática é a segunda fase do processo de compilação, onde os tokens gerados pela análise léxica são organizados em uma estrutura hierárquica que representa a sintaxe do programa. Esta fase verifica se o código-fonte está gramaticalmente correto de acordo com as regras da linguagem de programação.

—

A seção de "trabalhos relacionados"em um artigo de TCC (Trabalho de Conclusão de Curso) é onde o autor apresenta uma revisão da literatura existente relacionada ao tema de pesquisa do seu trabalho. Nesta seção, o autor geralmente discute estudos anteriores, pesquisas, teorias e descobertas relevantes que estão diretamente ligadas ao problema de pesquisa ou à questão que está sendo abordada no TCC.

O objetivo dessa seção é fornecer ao leitor uma compreensão mais ampla do contexto em que o trabalho está inserido, demonstrando o conhecimento existente sobre o assunto e destacando lacunas na literatura que justifiquem a realização do estudo em questão. Isso também ajuda o autor a posicionar seu trabalho em relação ao que já foi feito, destacando como ele contribui para o campo de estudo.

Normalmente, a seção de trabalhos relacionados inclui uma revisão crítica dos estudos existentes, identificando suas contribuições, limitações e possíveis áreas para futuras pesquisas. Isso ajuda a construir uma base sólida para o argumento do autor e a contextualizar a importância do seu próprio trabalho.

O referencial teórico, também conhecido como revisão teórica, é uma seção fundamental de um artigo de TCC (Trabalho de Conclusão de Curso). Nesta seção, o autor apresenta as teorias, conceitos e frameworks que sustentam a pesquisa realizada no trabalho.

O referencial teórico serve como a base conceitual e intelectual do estudo, fornecendo um arcabouço teórico que ajuda a fundamentar as hipóteses, justificar a metodologia escolhida e interpretar os resultados obtidos. Essa seção não apenas descreve as teorias relevantes para o tema do TCC, mas também mostra como essas teorias se relacionam com o problema de pesquisa específico abordado pelo autor.

3. Proposta

A natureza dinâmica da criação e sincronização de threads pode resultar em problemas de concorrência, como condições de corrida e deadlocks, que são difíceis de identificar e corrigir sem um entendimento profundo do modelo fork-join. Como resultado, muitos alunos enfrentam dificuldades ao desenvolver e depurar programas baseados neste paradigma.

O desenvolvimento de um compilador e plataforma com o foco no uso em sala de aula.

Será especificado uma gramática formal da linguagem de alto nível utilizada. A linguagem possuirá os primitivos Fork e Join como modelo de concorrência.

Será construído um analisador léxico, reportando aos alunos erros nos programas.

Será construído um analisador sintático, que modelara a estrutura do programa e reportará erros de construção.

Como saída esperada teremos, além das mensagens de erro ou de sucesso, um grafo de tarefas e dependências, além de métricas do programa especificado, como o caminho crítico e o grau máximo de concorrência.

Também será construído uma plataforma web, como um editor de textos integrado e renderização do grafo e métricas. Um exemplo de plataforma sendo o Compiler Explorer https://godbolt.org.

4. Metodologia

Será um pesquisa aplicada e exploratória.

Aplicaremos o protótipo em sala de aula, coletaremos dados e feedback dos alunos. O feedback será uma pesquisa qualitativa com as opiniões dos alunos sobre o uso do sistema em um formulário online.

O cronograma das atividades proposta está descrito abaixo.

4.1. Atividades

- **Atividade 1**: atividade 1;
- **Atividade 2**: atividade 2.

	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
Atividade 1	■					
Atividade 2		■				

5. Resultados esperados

Espera-se que a ferramenta proposta contribua significativamente para o aprimoramento do ensino de programação concorrente, fornecendo um recurso para o aprendizado dos alunos e auxiliando os professores no processo de ensino.

Referências

Conway, M. E. (1963). A multiprocessor system design. In *Proceedings of the November 12-14, 1963, fall joint computer conference*, pages 139–146.

Dennis, J. B. and Van Horn, E. C. (1966). Programming semantics for multiprogrammed computations. *Communications of the ACM*, 9(3):143–155.

Louden, K. (2004). *Compiladores - Princípios e Práticas*. Pioneira Thomson Learning.

Van Steen, M. and Tanenbaum, A. S. (2017). *Distributed systems*. Maarten van Steen Leiden, The Netherlands.