

FERRAMENTA EDUCACIONAL DE SUPORTE AO ENSINO DO MODELO *FORK-JOIN*

Trabalho de Conclusão de Curso

Discente: Lucas da Silva Marcos

Orientador: Prof. Dr. Rodolfo Adamshuk Silva

Universidade Tecnológica Federal do Paraná

Câmpus Dois Vizinhos

Bacharelado em Engenharia de Software

2025

Programação Concorrente

- ▶ Um programa concorrente é um programa que contém dois ou mais processos que trabalham juntos para realizar uma tarefa [Andrews 2001].

Contexto

- ▶ Programação concorrente: Execução simultânea de múltiplos processos.
- ▶ Importância: Essencial para o desenvolvimento de sistemas robustos e escaláveis.
- ▶ Desafios: Complexidade e abstração dos conceitos, falta de experiência dos estudantes.

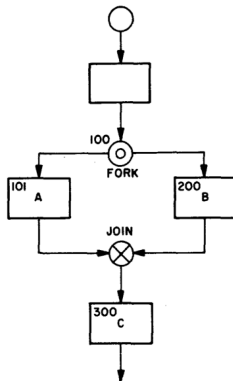
Programação Concorrente

- ▶ Programação concorrente permite a decomposição de tarefas complexas em subtarefas menores que podem ser executadas simultaneamente.
- ▶ Chaveamento de processos é a técnica que permite a alternância entre diferentes processos em execução.

Fork-Join

- ▶ Paradigma de programação concorrente que permite a decomposição de um problema em tarefas independentes que podem ser executadas em paralelo e, posteriormente, reunidas em um único resultado [Conway 1963, Dennis e Van Horn 1966].

Fork-Join



[Conway, 1963]

Fork-Join

- ▶ A sintaxe da instrução *FORK* (que realiza a criação de fluxos) é a seguinte:

```
FORK <rótulo>;
```

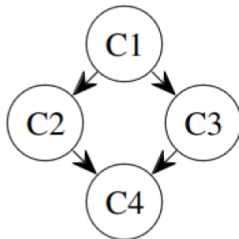
Fork-Join

- ▶ A sintaxe da instrução *JOIN* (que realiza a união de fluxos) é:

```
JOIN <variável-de-controle>, <rótulo-1>, <rótulo-2>;
```


Ensino do *Fork-Join*

► Pseudocódigo.



```
VAR_J = 2;  
C1;  
FORK ROT_C3;  
C2;  
JOIN VAR_J, ROT_C4, QUIT;  
  
ROT_C3:  
  C3;  
  JOIN VAR_J, ROT_C4, QUIT;  
  
ROT_C4:  
  C4;
```

Compiladores

- ▶ Compiladores são programas de computador que traduzem código-fonte de uma linguagem de programação para outra, geralmente de uma linguagem de alto nível para código de máquina [Louden 2004].

Compiladores

- ▶ Gramática.
- ▶ Análise Léxica.
- ▶ Análise Sintática.
- ▶ Análise Semântica.
- ▶ Geração de Código Executável.

Análise Léxica

- Converte o código-fonte em lexema, que são os elementos básicos da linguagem de programação.

```
FORK A;
```

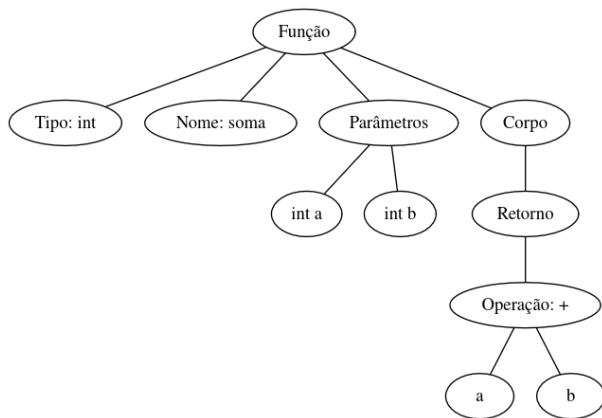
- * [Palavra Reservada, "FORK"]
- * [Identificador, "A"]
- * [Ponto e vírgula]

Análise Sintática

- Organiza os lexemas em uma estrutura hierárquica, verificando a correção gramatical do código-fonte.

Exemplo em C

```
int soma(int a, int b) { return a + b; }
```



Recursos Educacionais

- ▶ Recursos educacionais adequados, como ferramentas de desenvolvimento e exercícios práticos são essenciais para auxiliar os alunos no desenvolvimento de habilidades sólidas na área de programação.

CVisualizer

<pre>1 #include <stdbool.h> 2 #include <stdio.h> 3 #include <stdlib.h> 4 #include <string.h> 5 6 int recurse(int n) { 7 if (n > 0) { 8 int value = 2*n; 9 return value + recurse(n-1); 10 } 11 12 return 0; 13 } 14 15 int main() { 16 int a = 12; 17 for (int i = 0; i < a; i++) { 18 int a = 20; 19 20 recurse(i); 21 }</pre>	<div>stack ▾</div> <div>main() ▾</div> <div>a 12</div> <div>▾</div> <div>i 1</div> <div>▾</div> <div>a 20</div> <div>▾</div> <div>recurse(1) ▾</div> <div>n 1</div> <div>▾</div> <div>value 2</div> <div>▾</div> <div>recurse(0) ▾</div> <div>n 0</div>
---	---

[Rantanen, 2023]

Objetivo

- ▶ Desenvolver uma ferramenta educacional que facilite o ensino do modelo *fork-join*, utilizando conceitos de compiladores para verificar a sintaxe dos comandos e gerar o grafo de dependências.
- ▶ Auxiliar estudantes a compreenderem o modelo *fork-join*.
- ▶ Identificar erros sintáticos e lógicos no código.
- ▶ Visualizar o grafo de dependências gerado a partir do código inserido.

Ferramenta Web

- Criação de uma plataforma interativa que facilita o ensino e a aplicação prática do modelo *fork-join* em programação concorrente.

Requisitos Funcionais

- ▶ Editor de Código: Permite escrever e editar código *fork-join*.
- ▶ Análise Sintática em Tempo Real: Identificação imediata de erros.
- ▶ Geração de Grafo de Dependências: Representação visual das execuções paralelas.
- ▶ Detecção e Exibição de Erros: Mensagens claras para orientação do estudante.
- ▶ Sugestões de Correção: Ajuda a corrigir erros comuns.

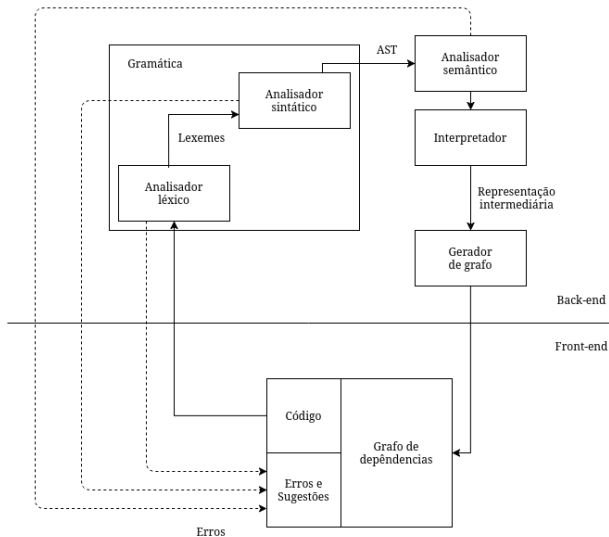
Requisitos Não Funcionais

- ▶ Desempenho: Atualizações em tempo real ($< 100\text{ms}$).
- ▶ Usabilidade: Interface intuitiva, adequada a diversos níveis de experiência.
- ▶ Compatibilidade: Suporte para diferentes navegadores e dispositivos.

Processo

<input type="checkbox"/>	<input checked="" type="radio"/> 23 Open <input checked="" type="radio"/> 17 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>	<input checked="" type="radio"/> FORKs and JOINs only on labels bug	#40 opened on Oct 16 by lucasmarcos					
<input type="checkbox"/>	<input checked="" type="radio"/> Recursive calls	#39 by lucasmarcos was closed 2 weeks ago					
<input type="checkbox"/>	<input checked="" type="radio"/> Implement SVG graph pan and zoom controls	#38 opened on Oct 14 by lucasmarcos					
<input type="checkbox"/>	<input checked="" type="radio"/> Linting: control variable of JOIN calls	#37 by lucasmarcos was closed 2 weeks ago					
<input type="checkbox"/>	<input checked="" type="radio"/> Loading screen. enhancement	#36 opened on Oct 3 by lucasmarcos					
<input type="checkbox"/>	<input checked="" type="radio"/> Greyout graph	#35 by lucasmarcos was closed on Oct 2					1
<input type="checkbox"/>	<input checked="" type="radio"/> Semantics: sequential call to a function already forked	#34 by lucasmarcos was closed 2 weeks ago					

Arquitetura



Front-end

- ▶ Editor de código (*CodeMirror*).
- ▶ Visualização de grafo (*Graphviz*).
- ▶ Painel de erros e sugestões.

Wireframe

x	x	x	x	
---	---	---	---	--

```
VAR_D = 2;  
A;  
FORK ROT_C;  
B;  
JOIN VAR_D, ROT_D, QUIT;
```

X Correção 1

X Correção 2

- Sugestão 1

- Sugestão 2

Ação
Ação
Ação
Ação

```
graph TD; A((A)) --> B((B)); B --> D((D)); D --> C((C)); C --> A;
```

+

.

Interface

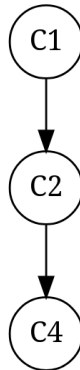
```
1 //  
2 // Bem-vindo à ferramenta educacional para aprendizado do modelo de  
3 // programação paralela fork-join! Esta é uma ferramenta interativa  
4 // onde você pode digitar seu código fork-join neste painel.  
5 // A ferramenta irá auxiliar você na correção e identificação de  
6 // erros comuns. No painel lateral, você poderá visualizar o  
7 // grafo de dependências do seu código, permitindo entender melhor  
8 // o fluxo de execução paralela e as relações entre as tarefas.  
9 //  
10  
11 VAR_J = 3;  
12  
13 C1;  
14 FORK ROT_C33;  
15 C2;  
16 JOIN VAR_J, ROT_C4, QUIT;  
17  
18 ROT_C3:  
19   C3;  
20   JOIN VAR_J, ROT_C4, QUIT;  
21  
22 ROT_C4:  
23   C4;
```

O valor da variável de controle não corresponde ao número de JOINS ✕

Corrigir

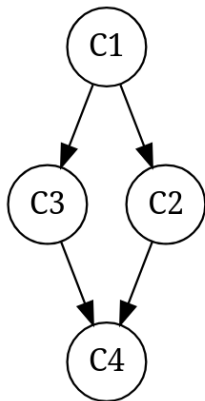
Chamada FORK para rótulo que não existe

Criar



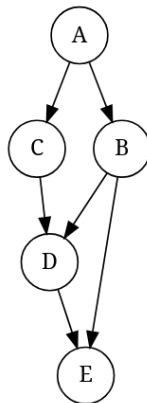
Boas-vindas

```
1 //
2 // Bem-vindo à ferramenta educacional para aprendizado do modelo de
3 // programação paralela fork-join! Esta é uma ferramenta interativa
4 // onde você pode digitar seu código fork-join neste painel.
5 // A ferramenta irá auxiliar você na correção e identificação de
6 // erros comuns. No painel lateral, você poderá visualizar o
7 // grafo de dependências do seu código, permitindo entender melhor
8 // o fluxo de execução paralela e as relações entre as tarefas.
9 //
10
11 VAR_J = 2;
12
13 C1;
14 FORK ROT_C3;
15 C2;
16 JOIN VAR_J, ROT_C4, QUIT;
17
18 ROT_C3:
19   C3;
20   JOIN VAR_J, ROT_C4, QUIT;
21
22 ROT_C4:
23   C4;
```



Erro

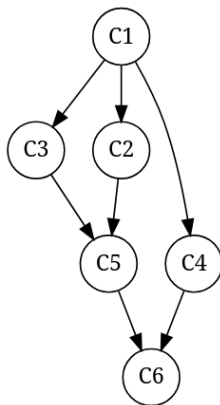
```
1  VAR_D = 1;  
2  VAR_E = 2;  
3  
4  A;  
5  FORK ROT_C;  
6  B;  
7  FORK ROT_E';  
8  JOIN VAR_D, ROT_D, QUIT;  
9  
10 ROT_C: C;  
11   JOIN VAR_D, ROT_D, QUIT;  
12  
13 ROT_D: D;  
14   JOIN VAR_E , ROT_E , QUIT ;  
15  
16 ROT_E': JOIN VAR_E, ROT_E, QUIT;  
17  
18 ROT_E: E;  
19   QUIT;
```



O valor da variável de controle não corresponde
ao número de JOINS [Corrigir](#)

Fluxos Distintos

```
1  VAR_C5=2;  
2  VAR_C6=2;  
3  
4  C1;  
5  FORK ROT_C3;  
6  FORK ROT_C4;  
7  C2;  
8  JOIN VAR_C5, ROT_C5, QUIT;  
9  
10 ROT_C3: C3;  
11   JOIN VAR_C5, ROT_C5, QUIT;  
12  
13 ROT_C4: C4;  
14   JOIN VAR_C6, ROT_C6, QUIT;  
15  
16 ROT_C5: C5;  
17   JOIN VAR_C6, ROT_C6, QUIT;  
18  
19 ROT_C6: C6;  
20   QUIT;
```



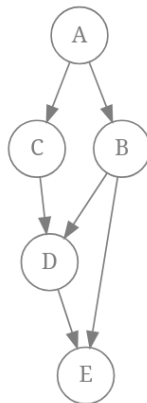
Ponto e vírgula

```
1  VAR_D = 1;  
2  VAR_E = 2;  
3  
4  A  
5  FORK ROT_C;  
6  B  
7  FORK ROT_E';  
8  JOIN VAR_D, ROT_D, QUIT;  
9  
10 ROT_C: C;  
11   JOIN VAR_D, ROT_D, QUIT;  
12  
13 ROT_D: D;  
14   JOIN VAR_E , ROT_E , QUIT ;  
15  
16 ROT_E': JOIN VAR_E, ROT_E, QUIT;  
17  
18 ROT_E: E;  
19   QUIT;
```

Faltando ponto e vírgula Adicionar

Faltando ponto e vírgula Adicionar

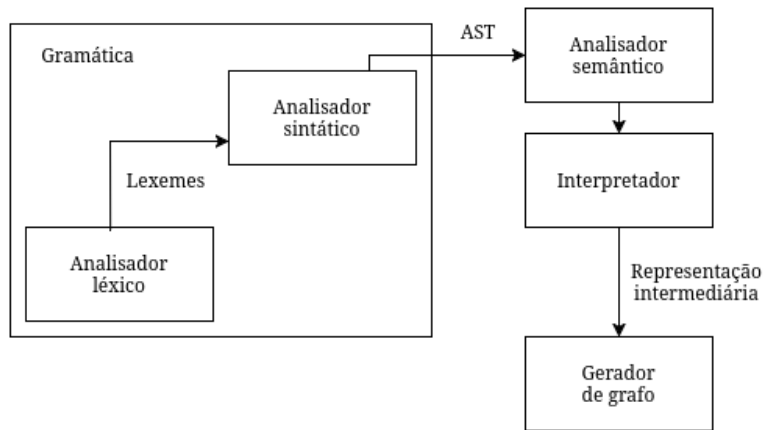
Faltando ponto e vírgula Adicionar



Back-end

- ▶ Análise sintática (*Tree-Sitter*) e semântica.
- ▶ Geração do grafo de dependências.

Back-end



Gramática

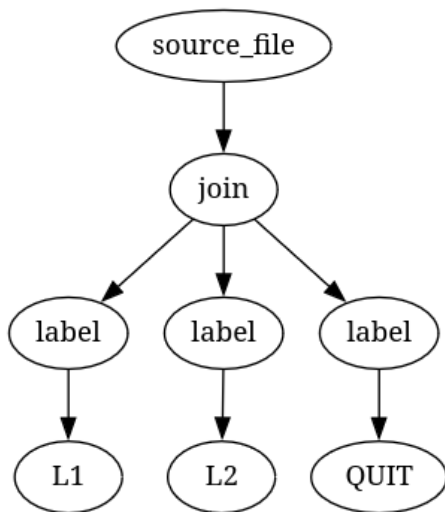
- ▶ Regras sintáticas específicas que governam a estrutura da linguagem de programação.
- ▶ Especificação Formal.

Análise

```
JOIN L1, L2, QUIT;
```

```
(source_file  
  (join  
    (label "L1")  
    (label "L2")  
    (label "QUIT"))))
```

Árvore Sintática Abstrata



Análise Semântica

- ▶ Verifica a correção lógica do código após a análise sintática.
- ▶ Identifica erros semânticos, como:
 - ▶ Uso de rótulos inexistentes em *FORK* ou *JOIN*.
 - ▶ Instruções inválidas após *QUIT*.
 - ▶ Variáveis definidas, mas não utilizadas.
- ▶ Garante que o código segue as regras do modelo *fork-join*.
- ▶ Gera sugestões de correção para o usuário.

Interpretador

- ▶ Executa o código passo a passo, analisando sua estrutura e comportamento.
- ▶ Simula a execução do modelo *fork-join*, garantindo que os fluxos sigam a lógica esperada.
- ▶ Identifica e sinaliza erros de execução.

Representação Intermediária

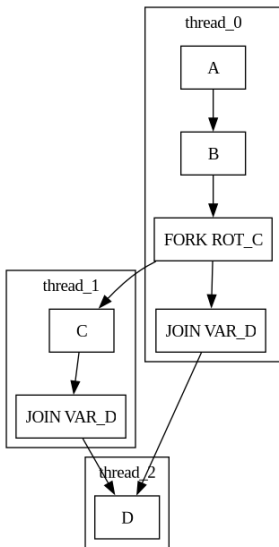
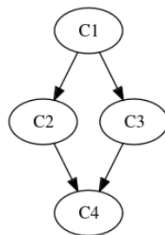


Tabela de Sugestões

Sugestão	Ação
Chamada <i>FORK</i> para rótulo que não existe	Criar
Chamada <i>JOIN</i> para rótulo que não existe	Criar
A última chamada em um <i>JOIN</i> deve ser <i>QUIT</i>	Trocar
Chamadas recursivas não são permitidas	Sem ação
Chamada de função após o <i>QUIT</i>	Remover
Variável de controle é definida e não utilizada	Remover
Falta de ponto e vírgula	Adicionar

Gerador de Grafo

```
digraph {  
    C1 -> C2;  
    C1 -> C3;  
    C2 -> C4;  
    C3 -> C4;  
}
```



Linguagem *DOT* do *Graphviz*.

Testes

- ▶ Cada requisito funcional foi associado a pelo menos um caso de teste.
- ▶ Construídos com base em exemplos.
- ▶ Validação da Sintaxe.
- ▶ Transformação em Grafo.
- ▶ Identificação de Erros.

Avaliação

- ▶ Aplicada em sala de aula com estudantes de Engenharia de Software.
- ▶ Coleta de opiniões via formulário.
- ▶ Melhora na identificação de erros e compreensão do modelo *fork-join*.

Questionário

- ▶ Formulário online aplicado a estudantes da disciplina de Programação Concorrente.
- ▶ Perguntas com escala *Likert* (Discordo Totalmente -> Concordo Totalmente).
- ▶ Opção de comentários para sugestões de melhorias.

Questionário

- ▶ Usabilidade
- ▶ Interface Gráfica
- ▶ Desempenho
- ▶ Aprendizado
- ▶ Identificação de Erros

Questionário

- ▶ Maioria dos estudantes considerou a ferramenta intuitiva e útil.
- ▶ A visualização em grafo ajudou na compreensão das relações de dependência.
- ▶ Sugestões para melhorias na detecção de erros.

Trabalhos Futuros

- ▶ Documentação.
- ▶ Extensibilidade.
- ▶ *Parbegin/Parend*.
- ▶ Acessibilidade.

Conclusão

- ▶ A ferramenta facilita o aprendizado de programação concorrente.
- ▶ Visualização de grafos e correção de erros ajudam na compreensão do *fork-join*.
- ▶ Potencial para evoluir e abranger novos paradigmas de concorrência.
- ▶ Relevância da programação concorrente no desenvolvimento de sistemas modernos.
- ▶ Necessidade de ferramentas educacionais para suporte ao ensino.

Referências

- ▶ Andrews, G. (2001). Foundations of Multithreaded, Parallel, and Distributed Programming. Addison-Wesley.
- ▶ 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. e Van Horn, E. C. (1966). Programming semantics for multiprogrammed computations. Communications of the ACM, 9(3):143–155.
- ▶ Rantanen, V. M. (2023). An Interactive C Code Execution and Visualization Tool for Online Learning (Master's thesis).

FERRAMENTA EDUCACIONAL DE SUPORTE AO ENSINO DO MODELO *FORK-JOIN*

Trabalho de Conclusão de Curso

Discente: Lucas da Silva Marcos

Orientador: Prof. Dr. Rodolfo Adamshuk Silva

Universidade Tecnológica Federal do Paraná

Câmpus Dois Vizinhos

Bacharelado em Engenharia de Software

2025