**Documento de Explicação do Código em Bash para a Disciplina de Compiladores**

---

Objetivo:

O script em Bash fornecido tem como objetivo compilar um arquivo fonte em C/C++ para bytecode LLVM e, em seguida, gerar grafos de fluxo de controle (CFG - Control Flow Graph) para cada função presente no arquivo fonte, em diferentes níveis de otimização. Este processo é útil para analisar como as otimizações de compilador afetam o fluxo de controle do programa.

---

Explicação:

1. Linhas 1-3: Verificação de Argumentos

- `if [ "$#" -ne 1 ]; then`: Verifica se o número de argumentos passados para o script é diferente de 1, ou seja, se o usuário não forneceu exatamente um argumento.
  - `echo "Uso: $0 <arquivo_fonte.c>"`: Exibe uma mensagem de uso do script, indicando ao usuário como deve ser utilizado.
  - `exit 1`: Encerra a execução do script com código de saída 1, indicando um erro.

2. Linhas 6-9: Verificação de Existência do Arquivo Fonte

- `ARQUIVO_FONTE="$1"`: Atribui o primeiro argumento passado para o script à variável `ARQUIVO_FONTE`, que representa o arquivo fonte.
- `if [ ! -f "$ARQUIVO_FONTE" ]; then`: Verifica se o arquivo fonte especificado não existe.
  - `echo "Erro: Arquivo fonte '$ARQUIVO_FONTE' não encontrado."`: Exibe uma mensagem de erro indicando que o arquivo fonte não foi encontrado.
  - `exit 1`: Encerra a execução do script com código de saída 1, indicando um erro.

3. Linhas 12-13: Definição dos Níveis de Otimização

- `NIVEIS_DE_OTIMIZACAO=("0" "1" "2" "3")`: Define um array `NIVEIS_DE_OTIMIZACAO` que contém os diferentes níveis de otimização que serão aplicados na compilação.

4. Linhas 16-23: Compilação do Arquivo Fonte e Geração dos Grafos de Fluxo de Controle

- `clang -emit-llvm -c "$ARQUIVO_FONTE" -o output.bc`: Compila o arquivo fonte em C/C++ para bytecode LLVM (`output.bc`) usando o compilador LLVM (`clang`) e gera o arquivo de saída com o nome `output.bc`.
- `for OTIMIZACAO in "${NIVEIS_DE_OTIMIZACAO[@]}"; do`: Inicia um loop sobre os diferentes níveis de otimização.
  - `llvm-dis -o output.ll output.bc`: Converte o bytecode LLVM (`output.bc`) em código LLVM legível (`output.ll`).
  - `opt -dot-cfg-only output.ll -o "cfg_$OTIMIZACAO.dot" -O$OTIMIZACAO`: Usa a ferramenta `opt` para gerar o CFG apenas para o código LLVM otimizado no nível atual (`$OTIMIZACAO`) e salva o resultado em um arquivo DOT (`cfg_$OTIMIZACAO.dot`).
  - `dot -Tpng -o "cfg_$OTIMIZACAO.png" "cfg_$OTIMIZACAO.dot"`: Converte o arquivo DOT em um gráfico PNG usando o utilitário `dot` e salva o arquivo com o nome `cfg_$OTIMIZACAO.png`.

5. Linhas 26-28: Limpeza de Arquivos Temporários

- `rm -f output.bc output.ll *.dot`: Remove os arquivos temporários gerados durante o processo de compilação e geração dos grafos de fluxo de controle.

6. Linha 30: Mensagem de Sucesso

- `echo "Grafos de Fluxo de Controle gerados com sucesso para o arquivo fonte '$ARQUIVO_FONTE'."`: Exibe uma mensagem indicando que os grafos de fluxo de controle foram gerados com sucesso para o arquivo fonte especificado.

---

Conclusão:

Este script automatiza o processo de compilação de um arquivo fonte em C/C++ para bytecode LLVM e a geração de grafos de fluxo de controle para cada função presente no código, em diferentes níveis de otimização. Esta análise é fundamental para entender o comportamento do programa e como as otimizações de compilador podem influenciar sua execução.