

A realização deste efolioB apresentou desafios significativos; permitiu aprender sobre a aplicação eficaz de multithreading para a divisão de tarefas entre várias threads, acelerando a pesquisa, especialmente em ficheiros grandes.

Passo agora a explicar Funcionamento Geral:

Leitura da Entrada:

O utilizador introduz a string de pesquisa, o nome do ficheiro e o número de threads a utilizar. A entrada é processada de forma segura para evitar erros e garantir a correta extração dos parâmetros.

- `strchr`: Encontra as aspas para delimitar a string de pesquisa.
- `strtok`: Divide a entrada em tokens (string, nome ficheiro, n° de threads).
- `strdup`: Aloca memória para a string de pesquisa e o nome do ficheiro.
- `atoi`: Converte o número de threads de string para inteiro

Divisão do Ficheiro:

O ficheiro é dividido em blocos, cada um atribuído a uma thread. Funções auxiliares garantem que cada segmento começa e termina nos limites de linhas, evitando que linhas sejam divididas entre threads.

- `find_start_line`: Calcula o número da linha inicial para cada thread, garantindo que começa no início de uma linha.
- `find_end_byte`: Calcula o byte de fim para cada thread, garantindo que termina no final de uma linha.

Pesquisa Paralela:

Cada thread procura a string de pesquisa no seu segmento. As linhas que contêm a string são armazenadas com os seus números de linha originais.

- `worker_task`: A função executada por cada thread.
- `getline`: Lê linha por linha do ficheiro.
- `strstr`: Procura a string de pesquisa na linha.

Recolha e Apresentação dos Resultados:

Os resultados de todas as threads são reunidos, ordenados e apresentados, mostrando o número e o conteúdo de cada linha encontrada.

- `global_line_count`: Imprime o número total de linhas encontradas.

Decisões de Implementação

Multithreading:

Uso da biblioteca POSIX threads (pthread_create e pthread_join) para criar e sincronizar as threads.

Sincronização:

Um mutex (pthread_mutex_t) protege a variável global que conta o número total de linhas encontradas, prevenindo condições de erro entre elas.

Estruturas de Dados:

Estruturas como TaskData e FoundLineInfo organizam os dados de cada tarefa e os resultados.

Gestão de Memória:

Alocação dinâmica com realloc e free permite guardar as linhas encontradas, mesmo sem saber previamente quantas serão.

Neste programa utilizei 2 variáveis globais: necessárias para a comunicação e sincronização eficientes entre as threads:

global_line_count: Esta variável armazena o número total de linhas que contêm a string de pesquisa. Ela deve ser global porque é acedida e modificada por várias threads. Cada thread incrementa esta variável quando encontra uma correspondência.

mutex: Este mutex protege o acesso à variável global_line_count. Como várias threads podem tentar atualizar esta variável simultaneamente, o mutex garante que apenas uma thread pode aceder a ela de cada vez, evitando a corrupção de dados.