Relatório: Ordenação de Doubles

Lucas Pires Camargo – 11103082
EMB5603 – Introdução às Estruturas de Dados

Introdução

Este trabalho visa a implementação de um programa para a ordenação de um arquivo contendo números de ponto flutuante double, com suporte a arquivos grandes (de 64KB até 512MB).

Implementação

O sistema de compilação escolhido foi o CMake, que gera makefiles customizados para o sistema em que o software está sendo compilado. O software foi dividido em quatro partes:

- `main.c` – Código principal, que executa a ordenação de acordo com os comandos do usuário, ou que executa o gerador de arquivos de entrada;
- `vec.h` – Uma biblioteca-header estática. Estrutura de dados que representa um vetor de doubles que se expande automaticamaente na memória. Possui funções para leitura e escrita em arquivos de texto.
- `buf.h` – Uma biblioteca-header estática. Estrutura de dados que representa um buffer de caracteres na memória. Possui funções de entrada e saída em arquivos. Utilizada para acelerar operações de entrada e saída.
- `mkinput.h` – Um gerador de arquivos de entrada de teste, incorporado ao programa.

O grafo de dependências do arquivo `main.c` mostrado na Figura 1 demostra como esses componentes se relacionam no código. Ele também mostra algumas bibliotecas do sistemas utilizadas.

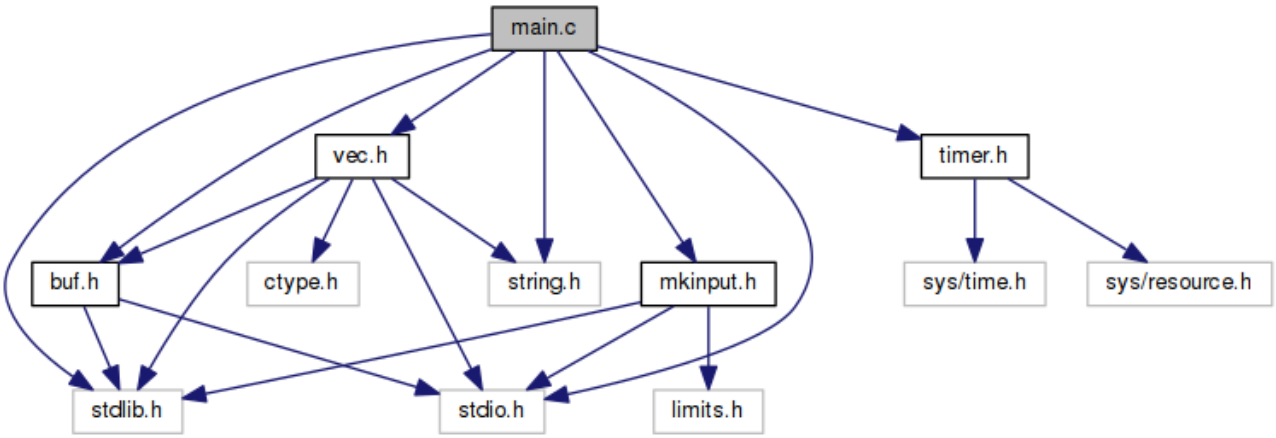

Figura 1: Grafo de dependências do arquivo `main.c`

A ordenação é feita por uma combinação de *mergesort* e *quicksort*. *Mergesort* é utilizado diretamente em listas muito grandes que possam causar problemas devido a níveis demasiadamente

elevados de recursão quando *quicksort* é utilizado. Se as sublistas tratadas pelo *mergesort* possuírem tamanho pequeno o suficiente, são ordenadas pelo *quicksort*. O limiar de tamanho da lista adotado no pragrama para essa decisão é de 20000 itens na lista. Essa abordagem adaptativa é atingida através do uso de uma função roteadora `vec_sort_smart`, que delega a ordenação ao algoritmo apropriado, e também passa adiante o buffer secundário utilizado no mergesort, para evitar alocação repetida de memória.

Testes Executados

Foram executados testes com arquivos de entrada de diversos tamanhos, e o programa foi instrumentado com funções do relógio monotônico fornecido pela chamada de sistema `gettimeofday` no Linux, e pelo *performace counter* no Windows. Os testes a seguir foram compilados pelo GCC 5.2.1 20151010 com otimização agressiva (*-O3)* e executados no Linux 4.2.0-18. O hardware apresenta um processador Intel Core(TM) i5-4210M CPU @ 2.60GHz e o sistema de arquivos é `ext4` em um SSD Crucial_CT250MX200SSD3. Os resultados são apresentados na Tabela 1.

Tamanho do Arquivo	Tempo de Entrada	Tempo de Ordenação	Tempo de Saída
64KB	0.0s	0.0s	0.15s
1MB	0.03s	0.03s	0.09s
100MB	0.88s	2.29s	6.59s
256M	2.16s	6.41s	17.62s
512MB	4.31s	14.09s	36.64s

Tabela 1: Resultados dos testes de ordenação

Pôde ser observado que os tempos de entrada e saída influenciaram o tempo de execução de maneira considerável, especialmente o tempo de escrita. Considerando apenas o tempo de ordenação, verificou-se que ele correspondeu ao tamanho do conjunto de dados de maneira adequada, e quase linear, para a faixa de tamanhos de arquivos de entrada que o software deve ser capaz de lidar.

Conclusão

Este trabalho foi importante para a validação de conhecimentos de programação básicos, gerenciamento de memória e leitura e escrita de terminal e em arquivos. Também foram exercitados algoritmos de ordenação e comparação.