Universidade do Minho

Laboratórios de Informática III

Relatório Fase 1 – Grupo 66

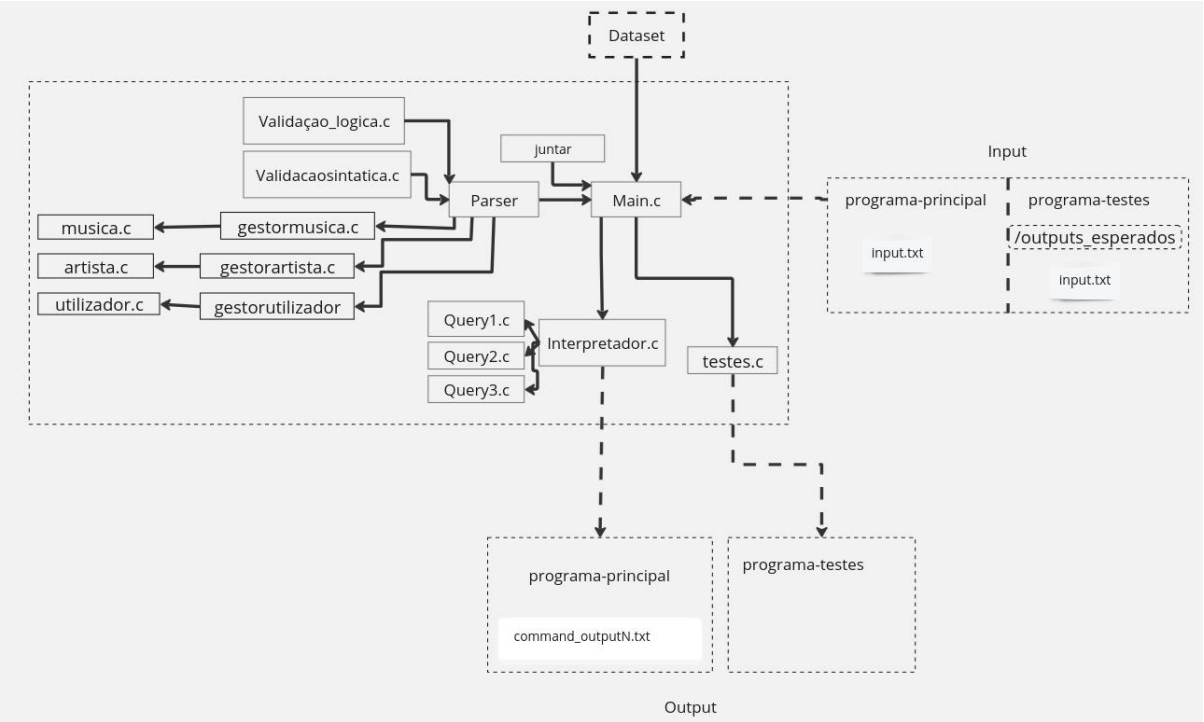
Introdução

Este relatório visa corresponder à análise do projeto no âmbito da disciplina de LI3. Este projeto (realizado em C) deve conseguir fazer o parsing de certos ficheiros csv e guardar a informação das entradas válidas destes ficheiros, que posteriormente serão usadas para responder às queries fornecidas.

Nesta primeira fase, o projeto deve:

- Fazer parsing dos ficheiros de entrada (musics.csv, users.csv, artists.csv);
- Validar estes ficheiros (cada entrada corresponda aos critérios dados);
- Ter um modo batch onde o terminal recebe três argumentos: o nome do executável, o caminho para a pasta onde estão os ficheiros csv de entrada e o caminho para o ficheiro de texto onde estão a lista de comandos a serem executados. Por exemplo, "./programa-principal ../dataset/data ../dataset/input.txt";
- Ler o ficheiro de comandos e executar todas as queries (3 no entretanto);
- Ter um programa de testes que avaliem os resultados obtidos, assim como o tempo de execução média e a memoria usada;

Sistema



Parser

O nosso parser recebe um inteiro que representa o ficheiro a ser recebido (0 para utilizadores, 1 para músicas e 2 para artistas). Depois disso, lê o ficheiro linha a linha, separando-o por campos delimitados por ";". Cada campo é validado e caso não

passe nas validações, essa linha é guardada no ficheiro de erros, senão vai para a nossa GHashTbale.

Validação dos utilizadores

Cada utilizador deverá ser constituído pelos seguintes campos:

- username – identificador único do utilizador;
- email – email de registo do utilizador;
- primeiro_nome– primeiro nome do utilizador;
- Ultimo_nome– apelido do utilizador;
- birth_date – data de nascimento;
- country – país onde a conta do utilizador foi registada;
- subscricao– tipo de subscrição, i.e., normal ou premium;
- Lista_gostadas– lista de identificadores únicos das músicas gostadas pelo utilizador.

Será necessário verificar:

- O formato do email (username@dominio),
- A subscrição apenas pode ser do tipo premium ou normal;
- O campo de listas_gostadas apenas pode ter IDs de músicas válidas;
- A birth_date tem de ser uma data válida no formato AAAA/MM/DD, não mais recente que a data atual(2024/09/09).

Validação das músicas

Cada música, para ser válida, tem de ter os seguintes campos:

- id – identificador único da música;
- title – nome da música;
- artist_id – lista de identificadores dos autores da música;
- duration – tempo de duração;
- genre – género da música;
- year – ano de lançamento;
- lyrics – letra da música.

Terá de ser também tido em conta:

- O campo artist_id de uma música, deverá corresponder a um artista existente e válido;
- A duração terá o formato HH:MM:SS e devem ser uma hora válida.

Validação dos artistas

Dado que cada entrada do artists.csv tem de ser uma string com os seguintes elementos:

- id – identificador único do artista;
- nome – nome do artista;
- description – detalhes do artista;
- recipe_per_stream – dinheiro auferido de cada vez que uma das músicas do artista é reproduzida;
- id_constituent – lista de identificadores únicos dos seus constituintes, no caso de se tratar de um artista coletivo. Este campo pode ser uma lista vazia;
- country – nacionalidade do artista;
- type – tipo de artista, i.e., individual ou grupo musical.

Será necessário verificar:

- O campo id_constituent de um artista individual não poderá ter elementos.

Implementação

Depois de serem realizados o parsing dos ficheiros de entrada, o programa abrirá o ficheiro de comandos dado como terceiro argumento no modo batch, em que irá ler cada linha e executar a query correspondente.

Cada linha executada deve criar um ficheiro txt do tipo commandN_output.txt, em que N varia consoante a linha do ficheiro de comandos que está a lida. No ficheiro commandN_output.txt, deverá ser inserido o resultado da execução da query.

Query 1

Comando: 1 <ID>

Output: email;first_name;last_name;age;country

Objetivo: Listar o resumo de um utilizador, consoante o identificador recebido por argumento.

Para a realização da Q1, o programa irá usar uma função predefinida nas GHashTables, em que encontra todas as informações de um user da GHashTable utilizador que tenham aquele ID. Como é garantido que o ID é único, não haverá nenhum problema no uso desta função.

Query 2

Comando: 2 <N> [country]

Output: name 1;type 1;discography duration 1;country 1

name 2;type 2;discography duration 2;country 2

Objetivo: Listar os N artistas com a maior discografia. Se o filtro país for usado, deve apenas retornar os N artistas com maior discografia daquele país.

Utilizamos um GArray para armazenar os dados que nos interessam, em que apenas guardamos os N maiores discografias. Caso apareça um melhor candidato dos que armazenados no array, adicionamos o mesmo descartando o com menor discografia armazenado.

Query 3

Comando: 3 <min age> <max age>

Output: genre 1;total likes

genre 2;total likes

Objetivo: Deve produzir como output uma lista ordenada de géneros por ordem decrescente de popularidade e o número total de likes associados.

A utilização de um array auxiliar com os generos de musicas possiveis facilita esta implementação. É percorrido todos os utilizadores de forma a encontrar todos entre as idades pretendidas. Quando isso ocorre, são percorridas todas as músicas a ver se existe alguma igual aos IDS apresentados na lista de músicas gostadas de cada utilizador selecionado. Caso aconteça, é procurado o genero dessa musica, de forma a adicionar um like ao array auxiliar.

Testes

1. **interpretador.c**:
    a. Este ficheiro contém a função `processar_comandos`, que lê comandos, identifica o tipo de query (Q1, Q2, Q3) e executa a função correspondente.
    b. Para cada query, o tempo de execução é medido com `clock_gettime`, e os resultados são guardados em ficheiros de saída na pasta `resultados`.
    c. No final do processo, calcula a média do tempo de execução para cada tipo de query e apresenta o valor.

2. **testes.c**:
    a. Este ficheiro verifica a correção dos outputs gerados comparando-os com os ficheiros esperados. A função `comparar_ficheiros` compara linha a linha cada ficheiro de saída com o esperado, indicando discrepâncias.
    b. A função principal `verificar_resultados` faz a contagem dos testes corretos para cada query.
    c. Mede também o tempo total de execução e o uso de memória do programa.

Relatório de Parâmetros

- **Tempo de Execução**:
    - Medido com `clock_gettime` tanto para cada query individual (em `interpretador.c`) como para o programa inteiro (em `testes.c`). A média de tempo é calculada e apresentada para cada tipo de query, enquanto o tempo total é exibido no final do programa.
    - Notar que usamos uma **flag** no **interpretador.c**, para conseguirmos distinguir o que é executado no **programa-principal** e **programa-testes,** sendo a **flag** igual a 0 e 1, respetivamente.
- **Memória Usada**:
    - Em `testes.c`, a função `getrusage` do cabeçalho `sys/resource.h` mede a memória máxima usada (`ru_maxrss`). O valor é impresso após a execução completa do programa.
- **Comparação de Outputs**:
    - A função `comparar_ficheiros` em `testes.c` compara os ficheiros de saída gerados pelo programa com os esperados (`commandN_output.txt`). Em caso de diferenças, a linha onde ocorre a primeira incongruência é indicada.

Desempenho

Tempo de execução:

Q1: 0.000004 s

Q2: 0.000000 s

Q3: 0.000520 s

Q1: 25 de 25 testes ok!

Q2: 25 de 25 testes ok!

Q3: 25 de 25 testes ok!

Elapsed time: 72.881718 s

Memory usage : 586260 KB

Conclusão

Ao longo deste projeto, explorámos e aplicámos diversas técnicas de programação em linguagem C. Através deste trabalho, conseguimos desenvolver uma base sólida de conceitos essenciais, como manipulação de dados, utilização de estruturas de controle e gestão de memória, aplicando-os na resolução dos problemas propostos.

Contudo, reconhecemos que existem áreas onde podemos e devemos melhorar. Em particular, iremos focar-nos em otimizar o código para maior eficiência e legibilidade, além de aprofundar a nossa compreensão sobre a gestão de erros e de memória. Esses aspetos serão essenciais para o desenvolvimento da segunda fase do projeto, onde pretendemos refinar e expandir as funcionalidades já implementadas, assegurando assim um trabalho de qualidade e que atenda aos critérios exigidos.

Estamos confiantes de que, com uma análise crítica do que realizámos até agora e com a implementação de melhorias, estaremos melhor preparados para enfrentar os desafios seguintes