USP - Universidade de São Paulo
Instituto de Ciências Matemáticas e Computação

Introdução à Ciência da Computação I
SCC0221

SISTEMA DE GERENCIAMENTO DE PLAYLISTS

1 Introdução

Recentemente, Juliana pediu sua ajuda para que construísse um gerenciador de músicas[1] em formato CLI (*Command Line Interface*). Na época, Juliana ainda era iniciante em programação então começou com um projeto simples: um gerenciador que manuseasse apenas uma playlist.

Estando com uma base mais sólida em programação, Juliana quer continuar seu projeto com um objetivo mais ambicioso: conseguir trabalhar com mais de uma playlist por vez, podendo armazená-las em arquivos a parte para que seus dados possam ser recuperados futuramente. Como você, estudante de computação, a ajudou em seu projeto passado, ela decidiu te chamar novamente para trabalhar neste projeto.

Estrutura dos arquivos de playlist

Os dados de uma determinada playlist, carregados em memória RAM, devem ser salvos em um arquivo de extensão *.dat* **em formato binário**. Esses arquivos possuem uma estrutura específica e constante, que pode ser observada visualmente a seguir:

Tabela 1: Cabeçalho de um arquivo de playlist

4 bytes	String	4 bytes
TamanhoNomePlaylist	NomePlaylist	NumMusicas

Tabela 2: Estrutura de um registro de música no arquivo de playlist

4 bytes	String	4 bytes	String	4 bytes	...
TamanhoNomeMusica1	NomeMusica1	TamanhoNomeArtista1	NomeArtista1	DuracaoMusica1	...

Em um arquivo binário, toda informação é escrita sequencialmente. Assim, o arquivo começa com a estrutura de cabeçalho (tabela 1) e depois com repetidos registros dos dados de música (tabela 2), de acordo com quantas músicas existem na playlist. Num arquivo binário **não existem** quebras de linha, os registros de cada música são escritos na ordem que foram inseridos na memória primária (RAM) e logo após o cabeçalho.

[1] O exercício 4, do bloco 2, construiu exatamente isso. Esse exercício pode ser enxergado como uma incrementação do exercício proposto anteriormente

- Nesse arquivo binário, o tamanho de cada string precede seu conteúdo. Isto é, **TamanhoNomePlaylist** se refere ao tamanho da string **NomePlaylist**, e assim sucessivamente.
- Entenda os 4 bytes como o valor de um inteiro.
- O caractere terminador da string (`'\0'`) **NÃO** é armazenado junto no arquivo de dados.
- **As tabelas foram construídas para propósitos plenamente visuais. No seu arquivo binário, as únicas coisas que serão armazenadas serão os conteúdos relativos as estruturas descritas nas tabelas, na mesma ordem**.
- Cada playlist é representada por um arquivo diferente. Não podem existir duas playlists em um mesmo arquivo.
- O ponteiro de música atual não deve ser armazenado no arquivo em nenhuma situação.

Sobre o gerenciador e suas funcionalidades

De maneira a inserir as novas funcionalidades no gerenciador, Juliana pretende mudar alguns dos códigos de operação das funcionalidades anteriores. **Repare que os comandos de 1 a 4 já foram implementados no exercício passado.**

- `Comando 1:` (*Já implementado*) Deve ser possível **adicionar uma nova música no final da playlist**, informando logo após as características que a fazem parte (nome, artista e tempo de duração, nessa ordem).
- `Comando 2:` (*Já implementado*) Deve ser possível **exibir na tela todas as músicas que estão contidas na playlist**, com uma formatação específica (vide seção 5). A música que está sendo tocada no momento também deve ser destacada.
- `Comando 3:` (*Já implementado*) Deve ser possível **avançar para a próxima música** da playlist.
- `Comando 4:` (*Já implementado*) Deve ser possível **retroceder para a música anterior** da playlist.
- `Comando 5:` Deve ser possível **salvar, em um arquivo binário, a playlist que está carregada atualmente na memória**, com o nome do arquivo de saída dado de entrada após a execução do comando.
- `Comando 6:` Deve ser possível **carregar uma playlist na memória RAM, advinda de um arquivo binário**. O nome do arquivo binário é dado de entrada após a execução do comando. Se já existir uma playlist carregada na memória RAM, essa deve ser descartada e substituída pela playlist do arquivo.
- `Comando 7:` Por fim, o usuário deve ter a opção de **finalizar o programa**.

Da mesma forma, cada comando é representado apenas por um número dado de entrada. As funcionalidades 5 e 6 **não devem comprometer** o funcionamento das demais, e deve ser possível adicionar novas músicas em playlists carregadas da memória secundária (disco).

Para uma playlist carregada a partir da memória secundária (disco), o ponteiro de música atual deve apontar para a primeira música da playlist, independentemente de onde o ponteiro estava na playlist antes da nova ser carregada.

2 Entrada

No começo da execução, será dado de entrada o nome da playlist. Logo após, serão dados de entrada indeterminados números relativos aos comandos já citados (até o programa receber o comando `7` para parar seu funcionamento). Cada comando sucederá dos argumentos que necessita (por exemplo, se dado o comando `1`, logo em seguida deverão ser lidas as características da nova música na ordem determinada anteriormente).

- O valor de duração de uma música é um número inteiro sem sinal.
- As funcionalidades 5 e 6 sucedem do respectivo nome do arquivo, em outra linha. Os nomes proporcionados já possuem a extensão *.dat*.
- Todos os nomes, artistas, nome de playlist e nome de arquivo são cadeias de caracteres *(strings)*. Todas elas possuem apenas caracteres presentes na tabela ASCII e podem conter mais de uma palavra.
- Nesse trabalho **não** há necessidade de tratamento de *strings* ou de valores de entrada. Essa garantia é assegurada pelas entradas que serão proporcionadas.
- Não há necessidade de tratar a inserção de músicas repetidas. Isso é assegurado pelas entradas que serão proporcionadas.
- A playlist não precisa ser circular. Em nenhum caso de teste a mudança na música tocada atualmente excede o limite da playlist por qualquer um dos lados.

3 Saída

As saídas permanecem da mesma forma que no exercício anterior, com excessão de que teremos agora novas mensagens para as funcionalidades 5 e 6, citadas abaixo.

- A cada nova música adicionada com sucesso, a mensagem '`Musica <nome_da_musica> de <nome_do_artista> adicionada com sucesso.\n`', onde os termos entre `<>` são relativos aos dados de entrada, deve ser exibida na tela.
- Caso a playlist esteja cheia, a mensagem '`Playlist cheia!\n`' deverá ser exibida ao usuário ao tentar adicionar uma nova música (e consequentemente exceder o número máximo) e a música não deve ser adicionada. O programa **não** deverá ser finalizado nessa situação.
- Quando a funcionalidade 5 (salvar playlist) for concluída com sucesso, a mensagem `'Playlist <nome> salva com sucesso.\n'` deve ser exibida na tela.
- Quando a funcionalidade 6 (carregar playlist) for concluída com sucesso, a mensagem `'Playlist <nome> carregada com sucesso.\n'` deve ser exibida na tela.
- Caso o arquivo dado de entrada nas funcionalidades 5 e 6 não exista, a mensagem `'Arquivo <nome_do_arquivo> nao existe.\n'` deve ser exibida e o programa deve ser finalizado.
- No final da execução de ambas as funcionalidades 5 e 6, a função `binaryToNum` (proporcionada em um arquivo a parte no `run.codes`) deve ser chamada para o arquivo correspondente que acabou de ser lido/escrito.

4 Orientações para confecção do código (Q&A)

(*As mesmas orientações do exercício passado também valem para esse!*)

1. Como prossigo com a abertura dos arquivos com nomes dados de entrada?

   - Utilize `"rb"` (*read binary*) e `"wb"` (*write binary*) como parâmetro da função `fopen` para abrir um arquivo para leitura e escrita, respectivamente.

2. Como posso realizar a leitura e escrita dos arquivos em questão?

   - Utilizando as funções `fread` e `fwrite`. Altamente recomendado que seja consultado o manual do Linux para visualização da documentação de ambas as funções (`'man 3 fread'` ou `'man 3 fwrite'`).

3. Os arquivos podem ser escritos/lidos com `fprintf` e `fscanf`?

   - Não. Aqui estamos trabalhando com arquivo de dados binário, que são orientados pelas funções citadas acima.

4. Como posso visualizar arquivos binários?

   - No shell do Linux, o comando `'hexdump -Cv <arquivo>'` é de grande ajuda para visualizar o conteúdo de arquivos binários (vide `'man 1 hexdump'`).

5. De que maneira consigo manipular as strings vindas de arquivo binário, uma vez que elas serão lidas **sem** o terminador (`'\0'`)?

   - Há diversas maneiras. Uma delas é inserir o caractere terminador após a leitura da string correspondente. A outra (mais recomendada) é utilizar a máscara `'%.*s'` da função `printf`. Essa máscara permite que você informe o número de caracteres da string a parte, de maneira que ele não passe dos limites da string (o tamanho da string é lida do arquivo de dados)

   Exemplo de utilização da máscara

   ```
   1  char *stringSemTerminador = "Hello!";
   2  int tamanhoString = 6;
   3  printf("%.*s", tamanhoString, stringSemTerminador);
   ```

6. Qual é a necessidade da função `binaryToNum`?

   - O `run.codes` não lida bem com a correção dos arquivos binários. Dessa forma, construímos uma função externa que interpreta o arquivo binário como um valor numérico real através de uma soma polinomial. O aluno não precisa entender o funcionamento da função, apenas se certifique de chamar a função **após fechar o arquivo** nas funcionalidades 5 e 6.
     Dessa forma, quanto mais próximo o valor numérico do valor esperado, mais semelhante ao arquivo binário esperado seu arquivo gerado está!

5 Exemplos de entrada e saída

- Exemplo 1: Salvando uma playlist em disco

Entrada

```
Amora e Manjericao
1
On Ira
Zaz
178
1
Comfortably Numb
Pink Floyd
382
1
Magic - Original Mix
Nhato
339
5
Amora_e_Manjericao.dat
7
```

Saída

```
Musica On Ira de Zaz adicionada com sucesso.
Musica Comfortably Numb de Pink Floyd adicionada com sucesso.
Musica Magic - Original Mix de Nhato adicionada com sucesso.
Playlist Amora_e_Manjericao.dat salva com sucesso.
4740.890000
```

- Exemplo 2: Carregamento de um arquivo inexistente

Entrada

```
Playlist de um arquivo inexistente
6
Nao_existe.dat
```

Saída

```
Arquivo Nao_existe.dat nao existe.
```

- Exemplo 3: Carregamento de uma playlist com outra já existente em memória

Entrada

```
Essa playlist sera removida
1
Makin' Waves
Vicarious Visions
175
6
Amora_e_Manjericao.dat
2
7
```

Saída

```
Musica Makin' Waves de Vicarious Visions adicionada com sucesso.
Playlist Amora_e_Manjericao.dat carregada com sucesso.
4740.890000
---- Playlist: Amora e Manjericao ----
Total de musicas: 3

=== NOW PLAYING ===
(1). 'On Ira'
Artista: Zaz
Duracao: 178 segundos

(2). 'Comfortably Numb'
Artista: Pink Floyd
Duracao: 382 segundos

(3). 'Magic - Original Mix'
Artista: Nhato
Duracao: 339 segundos
```

Bom trabalho! :)