

<p>ISCTE IUL Instituto Universitário de Lisboa Escola de Tecnologias e Arquitetura Dep. de Ciências e Tecnologias de Informação</p>	<p>Sistemas Operativos Ano letivo 2019/2020 1º Semestre versão 3</p>
--	---

Projeto Spotif-IUL (Parte 2)



Este enunciado detalha apenas as funcionalidades que devem ser implementadas na parte 2, sendo que os outros dois enunciados já foram ou serão publicados no decorrer do semestre.

A plataforma Spotif-IUL destina-se à partilha de músicas entre estudantes, docentes e colaboradores do ISCTE-IUL. Este programa (fictício!) pretende a estimular o estudo e o exercício físico. Para além de poderem ouvir as músicas preferidas, poderão também criar e partilhar playlists com os outros ouvintes.

Ouvinte: é a pessoa que deseja ouvir música e criar playlists.

Música: é uma música que o ouvinte pode escolher, caracterizada pelo seguinte conjunto de atributos: *identificador da música* (da forma *MUS_número*), *nome da música*, *artista*, *duração* (em segundos), *ano de produção*, *género*, *posição no top*.

Playlist: conterá a lista de playlists criadas pelos ouvintes, caracterizada pelo seguinte conjunto de atributos: *identificador da playlist* (da forma *PL_número*), *nome da playlist*, *músicas* (identificador de músicas da forma *MUS_número*).

Lyric: texto correspondente à letra de uma determinada música.

Os ouvintes registam-se na aplicação Spotif-IUL, escolhendo um *nickname* e uma *password* que depois usarão para poder ouvir as músicas. A aplicação também oferece a funcionalidade de criação e associação a playlists por parte do ouvinte, que poderão ser partilhadas com os colegas. Qualquer ouvinte se pode associar a uma playlist sem necessidade de qualquer autorização por parte do criador da playlist. As músicas podem estar associadas a uma letra de música (lyrics), que poderá ser visualizada ao mesmo tempo que a música toca.

Constituição dos grupos

O trabalho será realizado em grupos de 2 ou 3 estudantes (só em casos excepcionais podem ser aceites grupos de um estudante). Durante o semestre, alunos e professores poderão reorganizar os grupos por forma a evitar grupos de 1 estudante. Os grupos devem começar por se registar usando o e-learning:

- e-learning da UC Sistemas Operativos
- Escolher Menu [Grupos](#)
- Escolher um grupo disponível **que tenha o nome do seu docente das aulas práticas.**

A constituição dos grupos deverá manter-se até ao final do semestre, mas caso se verifiquem alterações, deve ser criado um novo grupo, mediante prévia autorização dos docentes, que passará a ser usado em vez do anterior para submeter os trabalhos.

Entrega, relatório e avaliação

A entrega da Parte 2 do trabalho será realizada através da criação de **um** ficheiro ZIP (**ATENÇÃO:** não serão aceites ficheiros RAR, 7Z ou outro formato) onde estarão todos os ficheiros criados **INCLUINDO** o relatório. O relatório será entregue no formato TXT ou PDF. Não serão aceites documentos DOCX ou outro formato. O ficheiro ZIP em questão terá OBRIGATORIAMENTE o nome do grupo em minúsculas, e.g., “so-cc-g01-parte2.zip” ou “so-ac-g23-parte2.zip”. Não serão aceites ficheiros com outros nomes que não o do grupo. Lembra-se que as iniciais do docente devem ser substituídas por aquelas a quem atribuiu o seu grupo:

- Prof. Carlos Coutinho – **cc**
- Prof. Ana Catarina Cruz – **ac**
- Prof. João Felício – **jf**
- Prof. Mário Rivotti – **mr**
- Prof. Paulo Pereira - **pp**

A entrega da Parte 2 do trabalho deverá ser feito por via eletrónica, através também do e-learning:

- e-learning da UC Sistemas Operativos
- Seleccionam o link do grupo onde estão inscritos
- Seleccionem o link “Trabalho Prático 2019/2020 Parte 2”
- Dentro do form “Visualizar Exercício de carregamento: Trabalho Prático 2019/2020 Parte 2”, seleccionem “Anexar Arquivo” e anexem o vosso ficheiro .zip. Podem submeter o vosso trabalho as vezes que desejarem, apenas a última submissão será contabilizada.

Observação. A submissão de um ficheiro substitui a versão anteriormente submetida. Realça-se que o formulário acima destina-se apenas à entrega da Parte 2.

Deverá ser criado um relatório em formato TXT ou PDF com informações sobre a elaboração do projeto. O relatório será entregue no formato TXT ou PDF. Não serão aceites documentos DOCX ou outro formato. Esse ficheiro deve chamar-se relatorio.txt ou relatorio.pdf, deverá ter aproximadamente 1 página A4 e deverá ser submetido dentro do ficheiro ZIP do trabalho.

O projeto será avaliado em termos globais produzindo-se uma nota de projeto, que será distribuída por todos os elementos do grupo, caso estes tenham todos contribuído de igual forma para o projeto. Caso existam diferenças no nível de contribuição, então os estudantes que mais contribuíram poderão ter uma nota individual superior à nota de projeto e o inverso para os estudantes que menos contribuíram. No fundo, parte da nota individual dos estudantes que menos contribuíram pode ser transferida para os estudantes que mais contribuíram, recompensando assim o esforço adicional de quem mais contribuiu. Assim, **o grupo deverá indicar no seu relatório uma estimativa de qual a percentagem de contribuição de cada elemento do grupo para o conjunto das tarefas realizadas (entre 0% e 100%), por forma a aferir qual foi a contribuição individual de cada elemento do grupo.** É ainda requerido uma justificação sucinta do porquê das diferenças nas percentagens atribuídas aos vários elementos do grupo (caso essas diferenças existam). Assume-se que a informação prestada é consensual entre os vários elementos do grupo. Quando isso não for possível, tal facto deverá ficar expresso no documento.

Política em caso de fraude

Os alunos podem partilhar e/ou trocar ideias entre si sobre os trabalhos e/ou resolução dos mesmos. No entanto, o trabalho entregue deve corresponder ao esforço individual de cada grupo. São consideradas fraudes as seguintes situações:

- Trabalho parcialmente copiado
- Facilitar a cópia através da partilha de ficheiros
- Utilizar material alheio sem referir a sua fonte.

Em caso de deteção de algum tipo de fraude, os trabalhos em questão não serão avaliados, sendo enviados à Comissão Pedagógica ou ao Conselho Pedagógico, consoante a gravidade da situação, que decidirão a sanção a aplicar aos alunos envolvidos. Serão utilizadas as ferramentas Moss e SafeAssign para deteção automática de cópias.

Recorda-se ainda que o Anexo I do Código de Conduta Académica, publicado a 25 de janeiro de 2016 em Diário da Republica, 2ª Série, nº 16, indica no seu ponto 2 que:

Quando um trabalho ou outro elemento de avaliação apresentar um nível de coincidência elevado com outros trabalhos (percentagem de coincidência com outras fontes reportada no relatório que o referido software produz), cabe ao docente da UC, orientador ou a qualquer elemento do júri, após a análise qualitativa desse relatório, e em caso de se confirmar a suspeita de plágio, desencadear o respetivo procedimento disciplinar, de acordo com o Regulamento Disciplinar de Discentes do ISCTE - Instituto Universitário de Lisboa, aprovado pela deliberação nº 2246/2010, de 6 de dezembro.

O ponto 2.1 desse mesmo anexo indica ainda que:

No âmbito do Regulamento Disciplinar de Discentes do ISCTE-IUL, são definidas as sanções disciplinares aplicáveis e os seus efeitos, podendo estas variar entre a advertência e a interdição da frequência de atividades escolares no ISCTE-IUL até cinco anos.

Parte 2 – Processos e Sinais

Data de entrega: 17 de novembro de 2019

Nesta parte do trabalho serão desenvolvidos os módulos **admin.c** (a correr no lado do servidor) e **player.c** (a correr no lado do cliente), utilizando a linguagem de programação C. No módulo Player, por motivos de copyright (e também para manter os nossos laboratórios livres de barulhos indesejados), não podemos disponibilizar os próprios ficheiros de música. Nesse sentido, vamos simular o tocar de uma música através da exibição de um gráfico com suporte a uma biblioteca fornecida.

Os programas a desenvolver irão operar com os ficheiros **musicas.txt** e **playlists.txt**, e incluirão ficheiros de texto do tipo ***.lyric** e ficheiros de dados binários. A estrutura de cada ficheiro é descrita em seguida.

Nota: Os ficheiros de texto referidos serão fornecidos pelos docentes na diretoria do servidor Tigre **“/home/so/trabalho-parte-2”**, pelo que não será necessária a criação ou alteração dos mesmos.

musicas.txt

Este ficheiro guarda a lista de músicas que existe na aplicação. Deve seguir o seguinte formato (é, na prática, o mesmo formato da parte 1, mas a duração agora está especificada em segundos e não em minutos):

ID_musica:nome_musica:intérprete:duração_em_seg:ano_lancamento:género:top

Exemplo:

```
MUS_1:Bohemian Rhapsody:Queen:359:1975:rock:40\n
MUS_2:Ai se ele cai:Xutos e Pontapes:191:2004:rock:27\n
MUS_3:We are the champions:Queen:190:1977:rock:1\n
MUS_4:Estou alem:Antonio Variacoes:303:1983:rock:13\n
MUS_5:Like a prayer:Madonna:338:1989:pop:2\n
```

playlists.txt

Este ficheiro guarda as playlists registadas no sistema, ou seja, a lista ordenada de músicas que pertence a cada playlist. Deve seguir o seguinte formato (é, na prática, o mesmo formato da parte 1):

ID_playlist:nome_playlist:lista_ordenada_de_identificadores_musicas

Exemplo:

```
PL_1:Jogging:MUS_3:MUS_2\n
PL_2:Estudo:MUS_5:MUS_3:MUS_4\n
PL_3:Classicas:MUS_8:MUS_10:MUS_12:MUS_1:MUS_15\n
```

<nome_musica>.lyric

Estes ficheiros são de texto e contêm letras de músicas. O próprio nome do ficheiro não é importante, mas sim o seu conteúdo: deve, obrigatoriamente, ser estruturado da seguinte forma:

- A primeira linha designa o nome da música;
- A segunda linha indica o nome do intérprete;

- Uma linha em branco;
- Uma ou mais linhas com a letra da música.

Exemplo:

Bohemian_Rhapsody.lyric

Bohemian Rhapsody\n

Queen\n

\n

Is this the real life? Is this just fantasy? \n

Caught in a landslide, no escape from reality\n

Open your eyes, look up to the skies and see\n

I'm just a poor boy, I need no sympathy\n

Because I'm easy come, easy go, little high, little low\n

Any way the wind blows doesn't really matter to me, to me\n

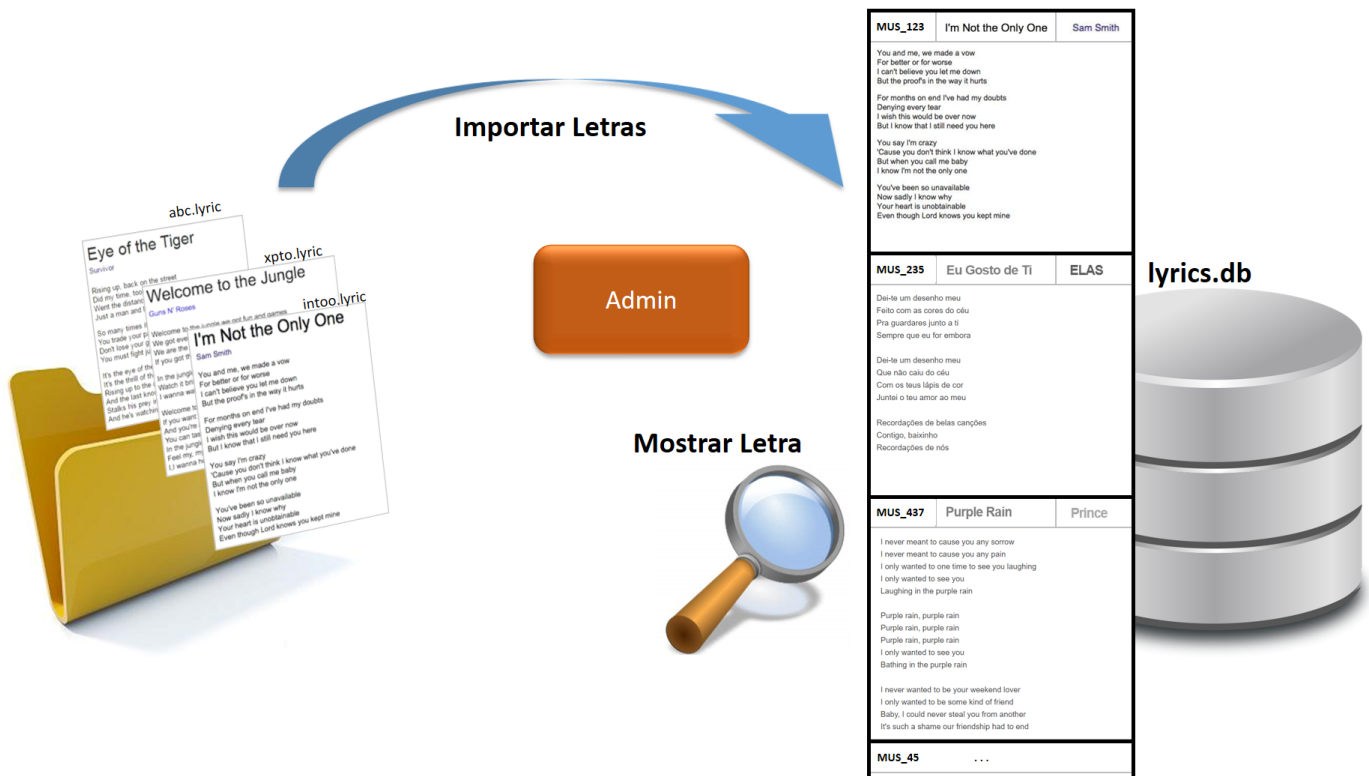
(...)

admin.c

O admin.c é o programa que, para já, vai ficar responsável por importar letras das músicas.

Deve começar por mostrar um menu ao administrador com duas opções:

- Importar letras
- Mostrar letra música



Nota: Não é suposto que no programa **admin.c** se carregue o ficheiro “**lyrics.db**” todo de uma vez para memória, os alunos deverão aceder a este ficheiro usando as funções de pesquisa de ficheiro (**fseek**) para percorrer as várias entradas neste ficheiro, com o mínimo desperdício de processamento.

Importar Letras

Na primeira opção, o programa pergunta ao utilizador para especificar uma diretoria onde estarão zero ou mais ficheiros contendo letras de música (ficheiros do tipo *.lyric). O objetivo desta opção é gerarem/atualizarem um único ficheiro na diretoria do admin, em formato binário, chamado “**lyrics.db**”, que contém todas as letras já importadas das diversas diretorias. Aquando da importação, as ações a realizar pelo **admin.c** serão as seguintes:

- Procurar, no ficheiro “**musicas.txt**” por uma entrada que tenha o título da música (primeira linha do ficheiro <musica>.lyric) e o intérprete da música (segunda linha do ficheiro <musica>.lyric);
- Se não encontrar a música no ficheiro “**musicas.txt**”, não importa o ficheiro e passa ao ficheiro seguinte, voltando ao ponto anterior;
- Se, pelo contrário, encontrar a música no ficheiro “**musicas.txt**”, guarda o seu identificador da música MUS_XXX;
- Procura no ficheiro “**lyrics.db**” uma entrada com o identificador da música encontrada, MUS_XXX;
- Se não encontrar nenhuma entrada no ficheiro “**lyrics.db**” com o identificador referido, cria uma nova entrada de registo no final do ficheiro “**lyrics.db**”;
- Se, pelo contrário, encontrar o ID da música no ficheiro “**lyrics.db**”, atualiza o campo Letra da correspondente entrada nesse ficheiro com o conteúdo lido do ficheiro lyric;
- Cada registo no ficheiro “**lyrics.db**” tem (pelo menos) a seguinte estrutura (fixa):
 - Identificação de Música (string com 10 caracteres no máximo) MUS_XXX;
 - Nome da Música (string com 60 caracteres no máximo);
 - Nome do Intérprete (string com 50 caracteres no máximo);
 - Duração da Música (inteiro com nº de segundos);
 - Letra (lyric) da música (string com 5000 caracteres no máximo).

Todos os ficheiros contidos na diretoria dada pelo utilizador devem ser lidos, mas só serão importados aqueles que tiverem um MUS_ID no ficheiro “**musicas.txt**”.

Mostrar Letras

Na segunda opção do menu do **admin.c**, o programa deve pedir ao utilizador um identificador de uma música (MUS_XXX), procurar o mesmo no ficheiro “**lyrics.db**” e mostrar a letra correspondente. Novamente, é suposto que esta pesquisa seja realizada com o mínimo de informação lida do ficheiro (tipicamente, apenas lendo os IDs das músicas no ficheiro “**lyrics.db**”).

player.c

O objetivo deste programa é fazer um *media player* de músicas. Para efeitos de manter o silêncio nos laboratórios e devido a questões de copyright das músicas, não será possível aqui reproduzir ficheiros de áudio. O objetivo passará então por simular uma interface para as operações básicas de um *media player* de música, PLAY, PAUSE, NEXT.

Para tal, o **player.c** deverá começar por pedir ao utilizador que indique o nome de uma playlist, e deve “carregar” as músicas correspondentes do ficheiro `playlists.txt` numa estrutura de dados de músicas (vetor ou lista). Após este carregamento, e assumindo que o Player terá à disposição um ficheiro com a mesma estrutura do “**lyrics.db**”, deverá “tocar” as músicas a começar pela primeira da lista etc. “Tocar” uma música significa usar a biblioteca de utilitários para mostrar o Media Player Spotify-IUL, mostrando as informações da música, e de acordo com a evolução da música, mostrar o progresso da mesma, a letra da mesma, e um gráfico de histograma.

Durante a reprodução da música, o utilizador poderá realizar as operações de pausa, retomar, avançar para a música seguinte, sair do *media player*. Todas estas operações deverão corresponder a uma operação no *media player*.

Para operar o Media Player Spotify-IUL terá à sua disposição as seguintes funções:

```
void startupSpotifIULMediaPlayer();
```



```
void showMediaPlayer(chtype color);
```

Estas funções iniciam o ambiente Spotif-IUL Media Player, e desenham um *media player* cuja “frame” terá a cor indicada no parâmetro.

```
void shutdownSpotifIULMediaPlayer();
```

Esta função termina o ambiente Spotif-IUL Media Player.

Os docentes colocaram na diretoria `/home/so/trabalho-parte-2/spotifiul-media-player` dois ficheiros chamado **example.c** e **example2.c** que poderão utilizar para verem provas de conceito da utilização da biblioteca.

Spotif-IUL v1.0	Music Information
<p>Music Player for ISCTE-IUL</p> <p>2019 (CopyLeft) Carlos Coutinho@iscte-iul.pt</p>	<p>Música: Don't Stop Me Now!</p> <p>Intérprete: Queen</p> <p>Progresso: </p>
Audio Visualizer	Music Lyrics
	<p>Tonight, I'm gonna have myself a real good time I feel alive and the world I'll turn it inside out, yeah And floating around in ecstasy So don't stop me now don't stop me 'Cause I'm havin</p>

Informação da Música:

```
void printSongInfo(char *songName, char *songSinger, chtype color);
```

Esta função escreve no Spotif-IUL Media Player a informação da música (nome, intérprete) indicados, na cor indicada.

Exemplo: `printSongInfo("Don't Stop Me Now", "Queen", IUL_COLOR_WHITE);`

Progress Bar:

```
void initProgress(int totalSeconds, chtype color);
```

Esta função cria a janela de progresso e escreve no Spotif-IUL Media Player a informação da duração da música em segundos, na cor indicada.

Exemplo: `initProgress(211, IUL_COLOR_CYAN);`

```
void setProgress(int currentSeconds);
```

Esta função escreve no Spotif-IUL Media Player a informação de progresso da música indicada.

Exemplo: `setProgress(82);`

```
void closeProgress();
```

Esta função fecha a janela de progresso, libertando a memória associada à mesma.

Gráfico de Histograma:

```
void initGraph();
```

Esta função cria a janela e inicia o gráfico de Histograma do Spotif-IUL Media Player.

```
void clearGraph();
```

Esta função “limpa” o gráfico de Histograma do Spotif-IUL Media Player.

```
void printGraph(int percent);
```

Esta função escreve um valor no gráfico de Histograma do Spotif-IUL Media Player. Para simplificar, apenas se irão considerar válidos valores entre 0 e 100.

Exemplo: `printGraph(67);`

```
void printRandomGraph();
```

Esta função escreve um valor aleatório (entre 0 e 100) no gráfico de Histograma.

```
void closeGraph();
```

Esta função fecha a janela de gráfico de Histograma, libertando a memória associada à mesma.

Letras de Música:

```
void initLyrics();
```

Esta função cria a janela de display de letras de música do Spotif-IUL Media Player.

```
void clearLyrics();
```

Esta função “limpa” a janela de letras de música do Spotif-IUL Media Player.

```
int printLyrics(chtype color, const char *format, ...);
```





Esta função escreve um texto na janela de Letras de Música do Spotif-IUL Media Player. A escrita deste texto não apaga nada e será realizada na cor indicada, e terá a mesma sintaxe do comando **printf()**.

Exemplo: `printLyrics(IUL_COLOR_LIGHTGRAY, "call me Mister Fahrenheit\nI'm traveling at");`

```
void closeLyrics();
```

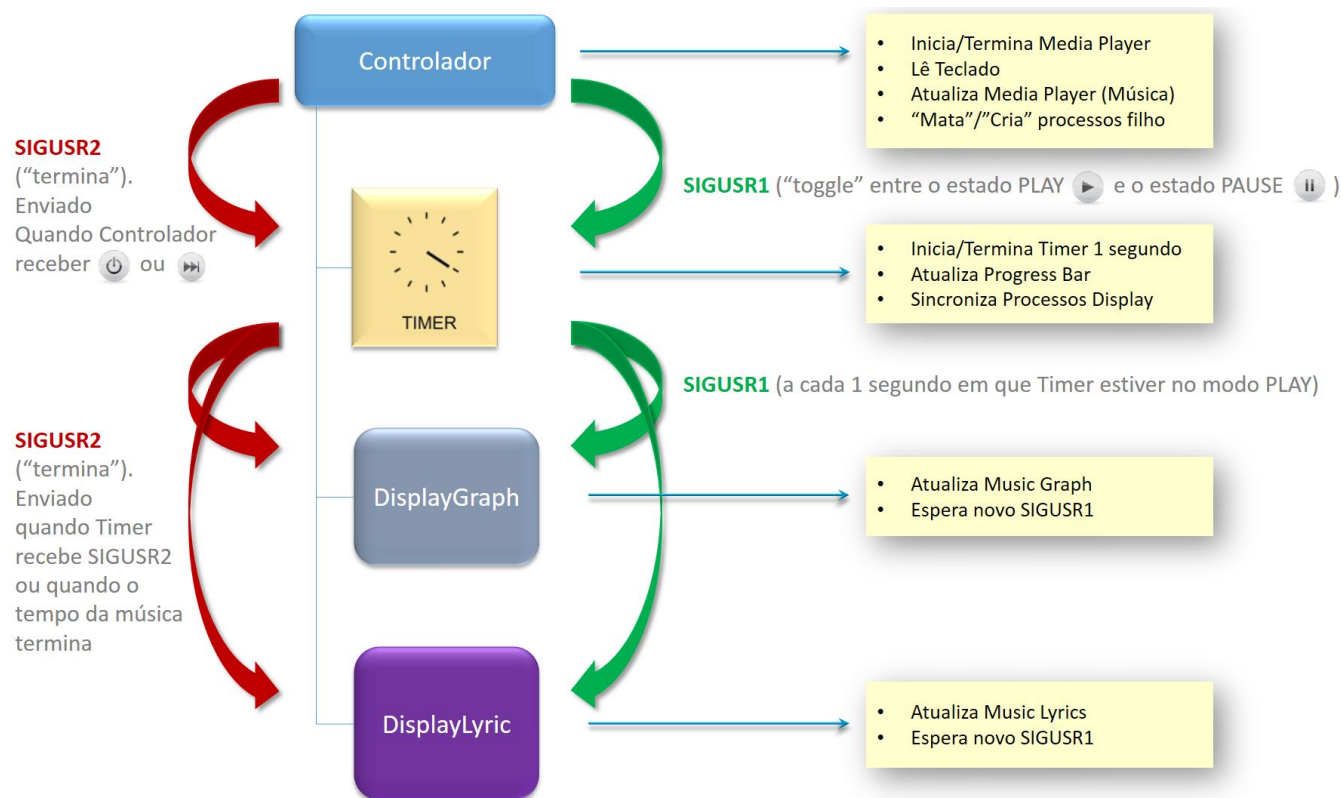
Esta função fecha a janela de letras de música, libertando a memória associada à mesma.

No **player.c**, o processamento do programa será controlado por quatro processos:

- **Processo Controlador (processo pai):** é aquele que inicia os outros processos (guardando os PID de todos eles para sincronização), e fica atento ao teclado à espera de comandos:
 - Se não receber nenhum comando, fica a aguardar o final de todos os processos “filhos” (correspondente ao final da música), após o que, passa para a próxima música da playlist;
 - Se receber o comando PAUSE  ou PLAY  envia um sinal **SIGUSR1** ao **Processo Timer** para que este alterne entre os estados PLAY e PAUSE;
 - Se for NEXT  envia um sinal **SIGUSR2** ao **Processo Timer**, para que este termine. Só depois de garantir que **todos** os processos “filhos” terminaram é que o processo “pai” cria novos processos “filhos” para tocarem a próxima música;
 - Se for QUIT  envia um sinal **SIGUSR2** ao **Processo Timer**, para que este termine. Só depois de garantir que **todos** os processos “filhos” terminaram é que o processo “pai” termina também, fechando o Media Player.
- **Processo Timer:** é criado pelo Processo Controlador, e quando nasce, cria a janela de Progresso, e a partir daí, de segundo em segundo, atualiza o progresso da música e envia um sinal **SIGUSR1** aos **Processos DisplayGraph** e **DisplayLyric**. Quando a música termina (ou seja, quando o tempo atual da música for igual ao tempo total da música), envia um sinal **SIGUSR2** aos **Processos DisplayGraph** e **DisplayLyric**, fecha a janela de progresso, e termina o seu próprio processo. O seu funcionamento varia também com os sinais recebidos pelo Processo Controlador:
 - Se receber um sinal **SIGUSR1** do Processo Controlador, pára o seu temporizador interno, deixando de atualizar o progresso da música e de enviar sinais aos outros processos; diz-se então que está no modo PAUSE. Se estiver neste modo PAUSE e receber novamente um sinal **SIGUSR1** do Processo Controlador, retoma o temporizador, passando novamente ao modo PLAY, e assim sucessivamente;
 - Se receber um sinal **SIGUSR2** do Processo Controlador, significa que a ordem é para terminar. Nesse caso, envia também um sinal **SIGUSR2** aos **Processos DisplayGraph** e **DisplayLyric**, fecha a janela de progresso, e termina o seu próprio processo.
- **Processo DisplayGraph:** é criado pelo Processo Controlador, e quando nasce, cria a janela de gráfico da música, e a partir daí, tem o seguinte comportamento:
 - Sempre que receber um sinal **SIGUSR1** (do Processo Timer), atualiza o gráfico da música;
 - Se receber um sinal **SIGUSR2** (do Processo Timer), significa que a ordem é para terminar. Nesse caso, fecha a janela de gráfico da música, e termina o seu próprio processo.
- **Processo DisplayLyric:** é criado pelo Processo Controlador, e quando nasce, cria a janela de letra da música, e a partir daí, tem o seguinte comportamento:
 - Sempre que receber um sinal **SIGUSR1** (do Processo Timer), escreve mais uma porção da letra da música, por forma a que quando a música terminar, ou pouco antes de terminar, tenha já sido mostrada toda a letra da música;
 - Se receber um sinal **SIGUSR2** (do Processo Timer), significa que a ordem é para terminar. Nesse caso, fecha a janela de letra da música, e termina o seu próprio processo.

Em nenhum caso é aceitável que qualquer dos processos esteja em espera ativa.

A figura seguinte mostra um esquema da interação entre os vários processos:



O Processo Controlador recebe os inputs possíveis, do teclado:



NEXT: Tecla 'N': Muda para próxima música da Playlist

- o programa avança para a próxima música mantendo o estado atual, PLAY ou PAUSE. Para tal, o Controlador **envia** um sinal do tipo **SIGUSR2** para o **Processo Timer**, "matando" os seus processos "filhos", e avança para a música seguinte da playlist, criando novos processos "filhos" para a música correspondente. Se estiver na última música da lista, e tentar avançar para uma próxima, deve voltar a apresentar a primeira da playlist. Quando iniciar uma nova música, o controlador tem que "carregar" toda a informação sobre a música (dados e lyric) e atualizar o Media Player com os dados da música.



PAUSE / PLAY: Tecla ' ' (espaço): Pára temporariamente o Media Player.

- Alterna entre os estados PLAY e PAUSE, **enviando** um sinal do tipo **SIGUSR1** para o **Processo Timer**, para que este pare o seu timer ou retome o seu timer.



QUIT: Tecla 'Q': Termina o Media Player (e toda a aplicação **player.c**).

- O Controlador **envia** um sinal do tipo **SIGUSR2** para o **Processo Timer**, "matando" os seus processos "filhos", e depois destes terminarem, fecha o Media Player e termina também.