Universidade do Minho

Processamento de Linguagens

Trabalho Prático I

Nelson Mota nº38573

Filipe Ribeiro nº64315

Abril de 2015

Conteúdo

1 Introdução

O enunciado escolhido para a realização deste trabalho pratico foi o 2.5 - Processamento de ficheiros com Canções. Neste projeto, será desenvolvido um processador que, recebendo um ou vários ficheiros contendo a letra das musicas bem como alguma meta-informação das mesmas, crie um ficheiro LaTeXpara cada uma, contendo informação da musica como o titulo e o autor, bem como a letra da mesma devidamente formatada.

Para este processador será necessário criar algumas estruturas de dados para suporte ao analisador léxico *Flex*, para isso foi utilizado a linguagem de programação C. Além das estruturas, utilizamos o Linguagem C para a criação dos ficheiros LaTeX.

2 Processamento de ficheiros com Canções

2.1 Estrutura dos ficheiros de canções

Cada ficheiro que contem a informação das musicas esta organizada por, inicialmente alguma meta-informação da musica em questão, seguido por a letra da mesma. Existe ainda a possibilidade de um ficheiro conter mais do que uma musica, neste caso, estão separadas por uma linha de hífens iniciada por "##".

2.2 Estruturas de Dados em C

2.2.1 Estrutura Musicas

Antes de desenvolver o analisador léxico, foi necessário criar estruturas de dados em C que nos possibilitasse guardar toda a informação obtida pelo mesmo.

Para guardar essa informação ficou decidido usar como estrutura uma lista ligada de musicas, com uma disciplina *LIFO*. Esta estrutura foi a escolhida pois acredita-se que é a mais indicada para este problema sendo também de fácil manuseamento.

Cada nodo da lista de musicas além de conter informação da musica em questão e sua respetiva letra, para conseguir formatar a letra da musica for criado um *array* de inteiros com uma tamanho definido como *TOTAL_ESTROFES*, de forma a que cada posição do mesmo *array* contenha o numero de vértices que constituem cada estrofe. Segue-se o esqueleto de estrutura implementada:

```
typedef struct musicas{
    char *title;
    char *author_lyric;
    char *author_music;
    char *author;
    char *from;
    char *singer;
    int *estrofes;
```

```
    struct letra *letra;
    struct musicas *next;
}*Musicas,Musica;
```

2.2.2 Estrutura Letra

Para conseguir guardar a letra de cada musica, optou-se por criar uma segunda estrutura para essa especifica função.

Tal como a estruturas principal, esta também implementada através de uma lista ligada de versos, mas, ao contrario da anterior esta segue uma disciplina *FIFO*, estando assim organizada por ordem de versos, desde o primeiro verso a cabeça da lista ao ultimo, na cauda da mesma. Sendo o esqueleto de estrutura letra implementada:

```
typedef struct letra{
    char *line;
    struct letra *next;
}*Letra;
```

2.3 Flex

Para processar o texto dos ficheiros que contem a informação das musicas, foi utilizado Expressões Regulares recorrendo ao *Flex*, que nos permitiu fazer *matching* e filtrar a informação pretendida. De forma a organizar a filtragem, recorremos à utilização dos contextos:

```
    %x musica letra
```

No contexto *musica* podemos apenas encontrar alguma meta-informação relativa a musica, após encontrar duas quebras de página passa par o próximo contexto *letra*, estando assim na letra da musica.

As expressões regulares utilizadas no *Flex* e as principais ações tomadas foram:

```
<*>(?i:title[ \t]*?:[ \t]*?).*    {
                    new = initMusicas();
                    mapalatex = iniciateMapaLatex();
                    setTitle(new,filterScharacters(yytext+7,p));
                    BEGIN musica;
                }
```

Em qualquer contexto, encontra a expressão que identifica o titulo da musica, inicia uma nova musica, carrega a lista de carácter especiais do latex, coloca o titulo na musica que criou e inicia o contexto musica.

`<musica>(?i:lyrics[ \t]*?:).*    { setLyric(new,yytext +8); }`
No contexto musica encontra o autor da letra.
`<musica>(?i:music[ \t]*?:).*    { setAutMusic(new,yytext+7) }`
encontra o compositor da musica.
`<musica>(?i:author[ \t]*?:).*    { setAuthor(new,yytext+8) }`
encontra o autor da musica.
`<musica>(?i:singer[ \t]*?:).*    { setSinger(new,yytext+8) }`
captura o cantor da musica.
`<musica>(?i:from[ \t]*?:).*    { setFrom(new,yytext+6) }`
encontra de quem é a musica.

Nas expressões referidas anteriormente ao fazer *matching* com algum dos casos, vai completando a estrutura que contem a musica atual.
`<musica>\n[ \t\n]+    { BEGIN letra; }`
no contexto musica, quando encontrar dois `\n` muda para o contexto letra que representa a letra da musica.

`<musica>(?i:[a-zA-Z]+:).*    { ; }`
qualquer outra meta-informação presente no cabeçalho da musica, não faz nada e avança no ficheiro.

```
<letra>"##".*    {
            setLetra(new,newLetra);
            ...
            musicas = insertMusic(musicas,new);
            BEGIN musica;
        }
```

quando no mesmo ficheiro faz *matching* com esta expressão, sinal que irá começar outra musica guarda a letra construida até ao momento atual, insere a musica na lista de musicas e volta ao contexto musica.

```
<letra>[^\n]*    {
    newLetra = insertLetra(newLetra,filterScharacters(yytext,p));
    ...
  }
```

No contexto letra, quando encontra uma linha terminada em `\n` adiciona mais a mesma linha a letra da musica, sendo que antes verifica a existência de algum carácter especial.

```
<*><<EOF>>    {
        ...
        setLetra(new,newLetra);
        musicas = insertMusic(musicas,new);
    }
```

Quando chegar ao final do ficheiro, executa o mesmo processo que executa quando encontra o separador de musicas, guarda a letra na musica correspondente e insere a musica na lista de musicas.

2.4 Makefile

O principal objetivo da Makefile é facilitar a compilação e execução do programa. Para isso criamos a seguinte ficheiro:

```
CC = gcc
CFLAGS = -O2
default: exec

clear:
    clear
clean:
    @rm -rf *.o lex.yy.c converted

lex.yy.c: readFile.l music_linkedlist.h makeLatex.h
    @flex readFile.l

prog: lex.yy.c music_linkedlist.c makeLatex.c
    @$(CC) $(CFLAGS) -o prog lex.yy.c music_linkedlist.c makeLatex.c -ll

exec: prog
    @cp prog IN
    @for i in `ls IN/*.lyr` ; do IN/prog "$$i"; done;

all: clear clean
```

Tirando partido de algumas funcionalidades do sistema UNIX executamos através de um ciclo *for* todos os ficheiros *.lyr* presentes na pasta IN, compilando-os todos de seguida, criando a uma nova pasta *converted* com todos os ficheiros *.tex* e seu respetivo PDF.

2.5 Gerador de LaTeX

Após guardar a informação das musicas na estrutura de dados, teremos que a tratar para poder ser representada num ficheiro LaTeXpara posteriormente ser compilado em *PDF*. Para isso foi criada um ficheiro em C que disponibiliza uma API,

```
void makelatex(Musicas musicas);
```

que se encarrega de tratamento da informação e criação do ficheiro LaTeXe sua compilação para *PDF*.

Este tratamento consiste, muito sucintamente, primeiramente na definição de varias *tags* obrigatórias na criação de um ficheiro LaTeX. Posteriormente para cada musica, vai construindo e acrescentando as *tags* necessárias para a formatação da mesma.

2.6 Exemplo de Utilização

2.6.1 Input / Output

```
title: Não é desgraça ser pobre
singer: Amália Rodrigues
author: Norberto Araújo
from: Javier Tamames
comm:  Fado menor do Porto

Não é desgraça ser pobre,
não é desgraça ser louca:
desgraça é trazer o fado
no coração e na boca.

Nesta vida desvairada,
ser feliz é coisa pouca.
Se as loucas não sentem nada,
não é desgraça ser louca.

Ao nascer trouxe uma estrela;
nela o destino traçado.
Não foi desgraça trazé-la:
desgraça é trazer o fado.

Desgraça é andar a gente
de tanto cantar, já rouca,
e o fado, teimosamente,
no coração e na boca.
```

Não é desgraça ser pobre

Norberto Araújo

Não é desgraça ser pobre,
não é desgraça ser louca:
desgraça é trazer o fado
no coração e na boca.

Nesta vida desvairada,
ser feliz é coisa pouca.
Se as loucas não sentem nada,
não é desgraça ser louca.

Ao nascer trouxe uma estrela;
nela o destino traçado.
Não foi desgraça trazé-la:
desgraça é trazer o fado.

Desgraça é andar a gente
de tanto cantar, já rouca,
e o fado, teimosamente,
no coração e na boca.

Amália Rodrigues

3 Conclusão

O principal objetivos deste trabalho e a aquisição e prática de conceitos de *Flex* bem como uma revisão de alguns conhecimentos de C (nomeadamente criação e manipulação das estruturas de dados).

A utilização do *Flex* revelou-se um verdadeiro desafio, pois, encontrar expressões regulares que satisfizessem todas as diferentes situações nos ficheiros de entrada não foi uma tarefa fácil.

Em termos da linguagem de programação C, a maior dificuldade terá sido chegar a um acordo na estrutura mais adequada a utilizar para este problema.

Por fim, a criação de dos ficheiros LaTeXapenas nos debatemos com a representação de caracteres especiais, tendo assim a necessidade substituir o carácter pela sua expressão para que este possa ser representado corretamente no ficheiro *PDF* de saída.

4 Apêndice

```
%x musica letra

%{
/* Declaracoes C diversas */
/*Fazer os includes */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "makeLatex.h"


//Inicializações
Musicas musicas = NULL;
Musicas new = NULL;
Letra newLetra = NULL;
Map mapaLatex = NULL;

int flag = 0;
int nVertices = 0;
int estrofe = 0;

char* aux;


%}


%%
<*>(?i:title[ \t]*?:[ \t]*?).*  { new = initMusicas();
                                  mapaLatex = iniateMapLatex();
                                  setTitle(new,filterScharacters(yytext+7,mapaLatex));
                                  BEGIN musica;
                                                          }
<musica>(?i:lyrics[ \t]*?:).*   {setLyric(new,yytext+8); }

<musica>(?i:music[ \t]*?:).*    { setAutMusic(new,yytext+7); }

<musica>(?i:author[ \t]*?:).*   { setAuthor(new,yytext +8); }

<musica>(?i:singer[ \t]*?:).*   { setSinger(new,yytext +8); }

<musica>(?i:from[ \t]*?:).*     { setFrom(new,yytext +6); }

<musica>\n[ \t\n]+              { BEGIN letra; }

<musica>\n                      ;

<musica>(?i:[a-zA-Z]+:).*       ;

<letra>"##".*                   {
                                 setLetra(new,newLetra);
                                 setEstrofe(new,estrofe,nVertices);
                                 nVertices = 0;
                                 estrofe = 0;
                                 flag = 0;

                                 newLetra = NULL;
                                 musicas = insertMusic(musicas,new);

                                 BEGIN musica;
                                }

<letra>[^\n]*                   {
                                 newLetra = insertLetra(newLetra,filterScharacters(yytext,mapaLatex));
                                 nVertices++;
                                 flag=1;
                                }
<letra>"\n"                     ;
```

```
<letra>"\n\n"                   {
                                 if(nVertices != 0 && flag == 1){
                                  setEstrofe(new,estrofe,nVertices);
                                  nVertices = 0;
                                  estrofe++;
                                 }
                                  flag = 0;
                                }

<*><<EOF>>                      {
                                 setEstrofe(new,estrofe,nVertices);
                                 setLetra(new,newLetra);
                                 musicas = insertMusic(musicas,new);
                                 yyterminate();
                                }
%%



int yywrap()
{ return(1);  }

int main(int argc, char **argv)
{
        ++argv, --argc;  /* skip over program name */
        if ( argc > 0 )
             yyin = fopen( argv[0], "r" );
        else
             yyin = stdin;

        yylex();

        makeLatex(musicas);
        fclose(yyin);
        return 0;
}
```

```c
#ifndef _MUSIC_LINKEDLIST
#define _MUSIC_LINKEDLIST



typedef struct letra{
  char *line;
  struct letra *next;
}*Letra,letras;


typedef struct musica{
  char *title;
  char *author_lyric;
  char *author_music;
  char *author;
  char *from;
  char *singer;
  int* estrofe;
  struct letra *letra;
  struct musica *next;
} *Musicas, Musica;



Musicas initMusicas();
Letra initLetra();


/*GET´S*/
Musicas getMusic(Musicas m, int indice);

/*INSERT´S*/
Musicas insertMusic(Musicas m, Musicas new);
Letra insertLetra(Letra l, char* v);

/*SET´S*/
void setTitle(Musicas m, char *t);
void setLyric(Musicas m, char *l);
void setAutMusic(Musicas m, char *a);
void setAuthor(Musicas m, char *a);
void setFrom(Musicas m, char *f);
void setSinger(Musicas m, char *s);
void setLetra(Musicas m, Letra l);
void setEstrofe(Musicas m, int i ,int n);

/*LENGHT'S*/
int lenghtMusicas(Musicas m);
int lenghtLetra(Letra m);

#endif
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "music_linkedlist.h"


Musicas initMusicas(){
        int i;
        Musicas aux = (Musicas)malloc(sizeof(Musica));
        aux->title = NULL;
        aux->author_lyric = NULL;
        aux->author_music = NULL;
        aux->author = NULL;
        aux->from = NULL;
        aux->singer = NULL;
        aux->estrofe=NULL;
        aux->letra = NULL;
        aux->next = NULL;
        return aux;
}

Letra initLetra(){
        return NULL;
}


void freeList(Letra l){
    if(l){
        if(l->next != NULL)
            freeList(l->next);
        free(l);
    }
}

void freeMusica(Musicas m){
    if(m){
        if(m->next != NULL)
            freeMusica(m->next);
        free(m);
    }
}

Letra copyLetra(Letra l)
{
    Letra new=initLetra();
    Letra aux = l;
    while(aux != NULL)
    {
        new = insertLetra(new, aux->line);
        aux = aux->next;
    }
    return new;
}


/*                                                                      *
** ADD´S                                                        *
**                                                                      */

/*Adicionar musica a lista de musicas */
Musicas insertMusic(Musicas m, Musicas new){
        if(m == NULL){
                m = new;
                m->next = NULL;
        }
        else{
```

```
                new->next = m;
                m = new;
        }
        return m;
}
/*Adicionar um verso a letra*/
Letra insertLetra(Letra l, char* v){
        if(l == NULL){
                l = (Letra)malloc(sizeof(letras));
                l->line = strdup(v);
                l->next = NULL;

        }
        else{
                Letra aux = NULL;
                aux = l;
                while(aux->next != NULL){
                        aux = aux->next;
                }
                Letra new = (Letra)malloc(sizeof(letras));
                new->line = strdup(v);
                new->next = NULL;
                aux->next = new;
        }
        return l;
}


/*                                                                      *
** Set´s para ir contruindo uma musica *
**                                                                     */

void setTitle(Musicas m, char *t){
        while(*t==' ') t++;
        m->title = strdup(t);
}
void setLyric(Musicas m, char *l){
        m->author_lyric = strdup(l);
}
void setAutMusic(Musicas m, char *a){
        m->author_music = strdup(a);
}
void setAuthor(Musicas m, char *a){
        m->author = strdup(a);
}
void setFrom(Musicas m, char *f){
        m->from = strdup(f);
}
void setSinger(Musicas m, char *s){
        m->singer = strdup(s);
}

void setLetra(Musicas m, Letra l){
        m->letra = (Letra)malloc(sizeof(letras));
        m->letra = l;
}


/*
** Array de inteiros que cada indice representa uma estrofe, em
   que representa o numero de vertices que a contituem                **/
/*i - nº da estrofe
* n - nº de versos
*/

void setEstrofe(Musicas m, int i ,int n){
        m->estrofe = realloc(m->estrofe,sizeof(int)*(i+1));
```

```
        m->estrofe[i]=n;
}

/*                                                                      *
** GET´S                                                        *
**                                                                      */


Musicas getMusic(Musicas m, int indice){
        Musicas res = (Musicas)malloc(sizeof(Musica));
        Musicas aux = (Musicas)malloc(sizeof(Musica));
        aux = m;
        int flag = 0;
        int j = 1;
        while(flag == 0){
                if(aux == NULL){
                        //out = 1;
                        return NULL;
                }
                else if(j == indice){
                        res = aux;
                        res->next = NULL;
                        flag = 1;
                }
                else{
                        aux = aux->next;
                        j++;
                }
        }
        return res;
}



/*                                                                      *
** LENGHT                                                       *
**                                                                      */

int lenghtMusicas(Musicas m){
  Musicas aux = m;
  int t= 0;
  while(aux != NULL){
    t++;
    aux = aux->next;
  }
  return t;
}
int lenghtLetra(Letra m){
  Letra aux = m;
  int t= 0;
  while(aux != NULL){
    t++;
    aux = aux->next;
  }
  return t;
}
```

```c
#ifndef _MAKELATEX
#define _MAKELATEX


#include "music_linkedlist.h"

typedef struct map{
        int count;
        char** simb;
        char** value;
}*Map,map;


void makeLatex(Musicas musicas);
char* filterScharacters(char * src, Map mapa);
Map insertRule(Map p, char* simb, char* expr);
Map iniateMapLatex();

#endif
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>

#include "makeLatex.h"



#define BEGIN_DOCUMENT "\\begin{document}\n"
#define END_DOCUMENT   "\\end{document}\n"
#define MAKETITLE      "\\maketitle\n"
#define NEW_LINE       "\\newline"



Map iniateMapLatex() {
  Map p = (Map) malloc(sizeof(Map));

  p->count=0;

  p = insertRule(p, "*" , "" );
  p = insertRule(p, "«" , "$\\ll$" );
  p = insertRule(p, "»" , "$\\gg$" );
  p = insertRule(p, "_" , "\\_" );
  p = insertRule(p, "#" , "\\#" );
  p = insertRule(p, "´" , "'" );
  p = insertRule(p, "[" , "(" );
  p = insertRule(p, "]" , ")" );
  p = insertRule(p, "-" , "" );

  return p;
}

Map insertRule(Map p, char* simb, char* expr) {
  if(p==NULL) {
    p = (Map) malloc(sizeof(Map));
    p->count=0;
  }
  (p->count)++;
  p->simb = (char**) realloc(p->simb,sizeof(char*)*(p->count));
  p->value = (char**) realloc(p->value,sizeof(char*)*(p->count));

  p->simb[p->count-1]=simb;
  p->value[p->count-1]=expr;

  return p;
}


char* filterScharacters(char * src, Map mapa) {

  char* aux = malloc(strlen(src)+1);
  memset(aux,'\0',strlen(src)+1);

  int n=0, i=0;
  while (*src!='\0') {
    int flag=0;
    for(i=0; i< mapa->count;i++) {

      if(strncmp(src,mapa->simb[i],strlen(mapa->simb[i]))==0) {
        int size=0;
        if(strlen(mapa->value[i])==0) {
          src+=strlen(mapa->simb[i])-1;
```

```
          n--;
          flag=1;
        } else {
          size=(strlen(src)+strlen(aux)+strlen(mapa->value[i])-strlen(mapa->simb[i])+1);

          //criação de buffer
          char* temp = NULL;
          temp=realloc(temp,size);
          memset(temp,'\0',size);

          //copia existente e valor novo
          strcpy(temp,aux);
          strcat(temp,mapa->value[i]);
          aux=temp;

          //incrementa posições
          n+=strlen(mapa->value[i])-1;
          src+=strlen(mapa->simb[i])-1;

          flag=1;
        }
      }
    }
    if(flag==0) aux[n]=*src;

    src++;n++;
  }
  return aux;
}



char * createTitle(char* title)
{
    char open[] = "\\title{";
    char close[] = "}";
    char * latextitle = malloc(strlen(title) + strlen(open) + strlen(close)+1);
    latextitle = strcpy(latextitle,open);
    latextitle = strcat(latextitle,title);
    latextitle = strcat(latextitle,close);
    return latextitle;
}


char * createAuthor1(char* author)
{
    char opentag[] = "\\author{";
    char closetag[] = "}";
    char* latexauthor = malloc(strlen(author) + strlen(opentag) + strlen(closetag)+1);

    latexauthor = strcpy(latexauthor,opentag);
    latexauthor = strcat(latexauthor,author);
    latexauthor = strcat(latexauthor,closetag);
  return latexauthor;
}

char * createAuthor2(char* authorL, char* authorM)
{

    char sep[] = "\\\\";

    char opentag[] = "\\author{";
    char closetag[] = "}";
    char* latexauthor = malloc(strlen(authorL) + strlen(authorL)+ strlen(opentag) + strlen(sep)
    +strlen(closetag)+1);
```

```
    latexauthor = strcpy(latexauthor,opentag);
    latexauthor = strcat(latexauthor,authorL);
    latexauthor = strcat(latexauthor,sep);
    latexauthor = strcat(latexauthor,authorM);
    latexauthor = strcat(latexauthor,closetag);
    return latexauthor;
}

char* listFrase(char* elem)
{
    char brk[] = "\\newline";
    char* listFrase = malloc(strlen(elem) + strlen(brk)+1);
    listFrase = strcpy(listFrase,elem);
    listFrase = strcat(listFrase,brk);
    return listFrase;
}

char * createSinger(char* singer)
{
    char opentag[] = "\\begin{flushright}";
    char closetag[] = "\\end{flushright}";
    char* latexsinger = malloc(strlen(singer) + strlen(opentag) + strlen(closetag)+1);

    latexsinger = strcpy(latexsinger,opentag);
    latexsinger = strcat(latexsinger,singer);
    latexsinger = strcat(latexsinger,closetag);
    return latexsinger;
}


char * initLatex()
{
  char init[] = "\\documentclass{article}\n\\usepackage[utf8]{inputenc}\n\\date{}\n\\raggedright\n";
  char* res = malloc(strlen(init)+1);
  res = strcpy(res,init);
  return res;
}

char * closeLatex()
{
  char close[] ="\\end{document}\n";
  char* res = malloc(strlen(close)+1);
  res = strcpy(res,close);
  return res;
}


void printMusica(Musicas music)
{
    printf("%s\n",createTitle(music->title));

    if(music->author_lyric != NULL && music->author!= NULL)
        printf("%s\n",createAuthor2(music->author_lyric,music->author));
    else{
        if(music->author_lyric != NULL)
            printf("%s\n",createAuthor1(music->author_lyric));
          else{
                printf("%s\n",createAuthor1(music->author));
          }
    }

    printf("%s\n",BEGIN_DOCUMENT);
    printf("%s\n",MAKETITLE);

    if(music->letra != NULL){
        Letra letra = music->letra;
        int i = 0;
```

```
        int v = 0;
        while(letra != NULL){
            v = music->estrofe[i];
          //  printf("%s\n",BEGIN_CENTER);
            while(v>0){
              printf("%s\n",listFrase(letra->line));
              v--;
              letra=letra->next;
            }
            //printf("%s\n",END_CENTER);
            i++;
            printf("%s\n",NEW_LINE);
        }
    }

    if(music->singer != NULL){
      printf("%s\n",createSinger(music->singer));
    }
}


void convertTex2Pdf (char *nome){
    char program[] = "pdflatex ";
    int size = strlen(program)+strlen(nome)+2+13;

    char * command = malloc(size);
    memset(command,'\0',size);


    strcpy(command,program);
    strcat(command,"\"");
    strcat(command,nome);
    strcat(command,"\"");
    strcat(command," 1>/dev/null");


    int i;
    i=system(command);
    i=system("rm *.log");
    i=system("rm *.aux");

    free(command);
}


void saveMusica(Musicas music)
{


  FILE*f = NULL;

  int size;
  char* nome = NULL;
  if  ((music->author != NULL)) {
    size = strlen(music->author)+strlen(music->title)+5+3;
    nome = malloc(size);
    strcpy(nome,music->author);
    strcat(nome," - ");
    strcat(nome, music->title);
  } else {
    size = strlen(music->title)+5;
    nome = malloc(size);
    strcpy(nome,music->title);
  }

  strcat(nome,".tex");
  f = fopen(nome, "w");
```

```
  if (f != NULL){
    fprintf(f,"%s\n",initLatex());
    fprintf(f,"%s\n",createTitle(music->title));

    if(music->author_lyric != NULL && music->author!= NULL)
        fprintf(f,"%s\n",createAuthor2(music->author_lyric,music->author));
    else{
        if(music->author_lyric != NULL)
            fprintf(f,"%s\n",createAuthor1(music->author_lyric));
          else{
                fprintf(f,"%s\n",createAuthor1(music->author));
          }
    }
    fprintf(f,"%s\n",BEGIN_DOCUMENT);
    fprintf(f,"%s\n",MAKETITLE);

    if(music->letra != NULL){
        Letra letra = music->letra;
        int i = 0;
        int v = 0;

        while(letra != NULL){
            v = music->estrofe[i];
            while(v>0){
              fprintf(f,"%s\n",listFrase(letra->line));
              v--;
              letra=letra->next;
            }
            i++;
            fprintf(f,"%s\n",NEW_LINE);
        }
    }




    if(music->singer != NULL){
      fprintf(f,"%s\n",createSinger(music->singer));
    }
    fprintf(f,"%s\n",END_DOCUMENT);
    fclose(f);
  }
  convertTex2Pdf(nome);
}


void makeLatex(Musicas musicas)
{
  char dir[] = "converted";
  int res;

  struct stat dirStat;
  if((res=stat(dir, &dirStat))<0)
      mkdir(dir, 0700);



  if(chdir(dir)!=0)
    printf("ERRO - Ocorreu erro ao mudar de directoria, não foi possivel efectuar conversão!\n");
  else {
    if(musicas->author != NULL)
      printf("Done! - %s - %s\n",musicas->author,musicas->title);
    else
      printf("Done! - %s\n",musicas->title);
    while(musicas != NULL)
    {
```

```
      saveMusica(musicas);
      musicas = musicas->next;
    }

  }
  if(chdir("..")!=0)
    printf("ERRO - Ocorreu erro ao mudar de directoria, verifique se a conversão foi efectuada!\n");

}
```