

Universidade do Minho
LEI 3º Ano 2º Semestre
Processamento de Linguagens
TP1

Daniel Caldas a67691
Marcelo Gonçalves a67736
Ricardo Silva a67728

1 de Abril de 2015

Resumo

Neste relatório são apresentadas propostas de solução para um conjunto de problemas selecionados pelo grupo, onde serão exploradas com o detalhe necessário as soluções implementadas, desde a análise de *datasets* até à explicitação do código fonte.

Conteúdo

1	Introdução	3
1.1	Estrutura do relatório	3
2	Museu da Pessoa	4
2.1	Análise/Especificação	4
2.1.1	Descrição informal do problema	4
2.1.2	Nota introdutória	4
2.1.3	Estrutura do site	5
2.1.4	Estratégia	6
2.2	Requisitos	6
2.2.1	Análise dos <i>datasets</i>	6
2.3	Filtros de texto	8
2.3.1	Condições de contexto	8
2.4	Concepção	10
2.4.1	Estruturas de dados	10
2.4.2	Algoritmos	11
2.4.3	Módulo em C	11
2.5	Problemas	12
2.6	Shell Scripting	14
2.7	criarhome.c	14
2.8	Alternativas	14
2.9	Testes efetuados	15
2.10	Resultados	16
3	Ficheiros com Canções	19
3.1	Análise e Especificação	19
3.1.1	Descrição informal do problema	19
3.1.2	Especificação dos Requisitos	19
3.2	Concepção	20
3.2.1	Estruturas de Dados	20
3.2.2	Estratégia	21
3.3	Codificação	21
3.3.1	Implementação	21
3.3.2	Testes realizados	22

3.3.3	Resultados	23
4	Enamex	25
4.1	Nota	25
4.2	Análise e especificação	25
4.2.1	Descrição informal do problema	25
4.2.2	Especificação dos requisitos	25
4.3	Filtros de texto	26
4.4	Extrair conteúdo	26
4.5	Testes e Resultados	27
5	Conclusão geral	28
5.1	Aprendizagem	28
5.2	Análise Crítica geral	28
A	Código do Programa 2.1	29
A.0.1	filtro.l	29
A.0.2	gerealbum.c	35
A.0.3	gerealbum.h	45
A.0.4	makefile	45
A.0.5	criarhome.c	45
A.0.6	run.sh	47
A.0.7	clear.sh	48
A.0.8	Ficheiros Input	48
A.0.9	Ficheiros Output	54
B	Código do Programa 2.5	59
B.0.10	filtrocancoes.l	59
B.0.11	List.c	64
B.0.12	List.h	67
B.0.13	runcancoes.sh	69
B.0.14	clearcancoes.sh	69
C	Código do Programa 2.2	70
C.0.15	enamex.l	70

Capítulo 1

Introdução

No contexto da UC Processamento de Linguagens foram propostos uma série de enunciados práticos para o desenvolvimento de filtros de texto usando como ferramenta o ***FLEX***, armazenando os dados obtidos em estruturas de dados escritas na linguagem **C** para que a mesma informação possa ser tratada e posteriormente apresentada em variados formatos.

O nosso objetivo será então implementar soluções para os exercícios **2.1 Museu da Pessoa - tratamento de fotografias** e **2.5 Processamento de ficheiros com Canções**. Abordaremos também ligeiramente o exercício **2.2 - Processamento de Entidades Nomeadas (Enamex)**.

1.1 Estrutura do relatório

Neste relatório vamos expor propostas de solução para os problemas 2.1, 2.5 e 2.2 (solução parcial), seguindo para ambos os casos a ordem dos pontos¹:

- Estratégia/Abordagem ao problema;
- Análise dos *datasets*;
- Explicação dos filtros de texto criados;
- Estruturas de dados e módulos em C;
- Testes e análise de resultados;

¹A estrutura pode eventualmente sofrer alterações mediante as especificidades de cada um dos problemas.

Capítulo 2

Museu da Pessoa

2.1 Análise/Especificação

2.1.1 Descrição informal do problema

É fornecido um conjunto de ficheiros xml resultantes de entrevistas, estes ficheiros são meta-dados de albuns fotográficos em que cada fotografia contém inúmeras anotações sobre as pessoas, a data, o local etc.

Basicamente o que é pedido, é que se crie um álbum HTML com todos esses dados e fotografias ordenados cronologicamente, isto após uma análise rigorosa dos *data-sets*.

2.1.2 Nota introdutória

A solução implementada para este problema é a criação do ***site* Museu da Pessoa, que agrega um conjunto de *datasets*** disponibilizados pela equipa docente. Para melhor compreensão do trabalho desenvolvido, antes de qualquer explicação detalhada é feita uma abordagem *top-down* do problema, isto é, vamos primeiro dar a conhecer a arquitetura do site para que seja mais fácil de compreender o objetivo desta nossa **solução**.

2.1.3 Estrutura do site

Por forma a tornar apresentável a informação dispersa pelos diversos ficheiros *.xml*, o grupo de trabalho teve primeiro o cuidado de pensar como iria ser exposta a informação tratada. Decidiu-se então por uma simples estrutura que consiste no seguinte:

- **Página inicial:** Uma página a partir da qual poderemos aceder aos álbuns gerados por um determinado *dataset*. Por exemplo no caso concreto da **Taberna do Fausto**, o *dataset* terá uma própria página onde serão dadas a conhecer as diversas pessoas e respetivos álbuns de fotografias.
- **Página de *dataset*:** Nesta página poderemos aceder aos álbuns de pessoas referenciadas dentro de um dado *dataset*.
- **Álbum de fotos:** Para cada pessoa referenciada será criado o respetivo álbum de fotografias.

Esta estrutura acabada de descrever, está bem ilustrada de forma **genérica** na seguinte **árvore de ficheiros**.

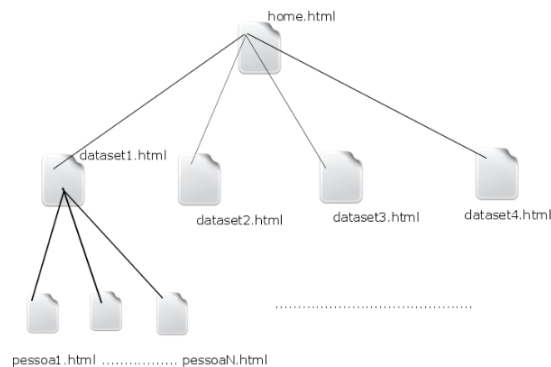


Figura 2.1: Estrutura do nosso site Museu da Pessoa, um pequeno *sistema de ficheiros*, **visão global**

2.1.4 Estratégia

Com base na ideia da secção anterior, o próximo passo do grupo foi delinear uma estratégia para a resolução do problema. Seguimos então os seguintes passos:

- Escrever em **HTML** templates para as diversas páginas, por forma ser mais simples de implementar funções que alterem conteúdo **dinamicamente**;
- Análise detalhada dos *datasets* disponibilizados por forma a construirmos filtros de texto que captassem toda a informação necessária;
- Implementação do filtro de texto em **FLEX**, implementando também algumas funções auxiliares na linguagem C;
- Desenho de uma estrutura de dados que fosse adequada para o problema;
- Implementação de um módulo em C capaz de gerir os dados passados pelo filtro e armazenando-os na estrutura de dados;
- Realização de alguns testes acerca do conteúdo filtrado e de dados armazenados;
- Completar o módulo em C com funções que geram automaticamente as páginas **HTML**;
- Por último criação de **shell scripts** que permitem correr o programa dentro de todos os *datasets* agrupando assim toda a informação, num só site.

2.2 Requisitos

2.2.1 Análise dos *datasets*

Fotografias

Em cada pasta disponibilizada reparamos que existem imagens, com um formato do nome muito específico:

XXX-Y-XX.YYY,

em que **X** são dígitos e **Y** uma letra (normalmente maiúscula). Estas imagens dos *datasets* encontravam-se nos formatos **jpg** e **gif**.

Datas

O formato das datas não se apresenta tão consistente como o das fotos, pelo que tivemos tal facto em conta para a escrita de filtros **flexíveis** que conseguissem "suportar" os diversos formatos:

- `<quando data="AAAA-MM-DD"/>`;
- `<quando data="AAAA-MM-DD">`, o mesmo que o anterior mas sem a marca de fecho;
- `<quando>AAAA</quando>`, em que não existe o atributo `data` na marca `<quando>` mas sim um valor que frequentemente representa o ano.;

Nomes

Para identificar uma pessoa ou conjunto de pessoas identificadas numa foto é usada a marca `<quem>` seguindo seguinte formato:

`<quem>NNN</quem>`,

em que `NNN` será um qualquer conjunto de caracteres que forma o nome de uma pessoa.

Ocorrência do carater de mudança de linha

Por vezes existem caracteres de mudança de linha no meio destas marcas xml, a estratégia adoptada foi apanhar a marca de abertura e fecho separadamente (caso a ocorrência de mudança de linha) e concatenar o conteúdo filtrado com auxílio da primitiva do C `strcat`.¹

Facto

O conteúdo filtrado desta marca representa o título de uma foto, este tal como a marca `<quem>` surge ocasionalmente "fragmentado", pelo que é adoptada a mesma estratégia já explicada.

Legenda

Pelo menos uma vez surge a marca `<legenda>` nos ficheiros. A filtragem desta marca resulta já de uma fase de optimização do programa. Colocamos a legenda por baixo da respetiva imagem.

Local

Durante a análise dos *datasets* surgiu também a marca *local* (que assumimos que ser o local onde foi tirada a foto). Esta funcionalidade também é extra como a anterior, no sentido em que é pedido no enunciado.

¹Esta estratégia será adoptada noutras situações, será feita referência a esta secção por forma a não repetir a informação.

2.3 Filtros de texto

Com base na análise feita na secção anterior desenvolvemos os vários filtros de texto para tratamento dos ficheiros *xml* e fotografias, vamos de seguida explicitar dois dos filtros que demonstram a generalidade do que foi implementado ao nível da ferramenta **FLEX**.

```
\<quando{espaco}?(data{espaco}?\"{data}\")?(\\/)?\">(.*\</quando\>)? \\
```

O filtro para captar datas é muito versátil pelo que consegue filtrar as diversas **mutações** que esta marca *<quando>* pode apresentar, de salientar:

- `(\\/)?>`

Permite opcionalidade na marca de fecho "interna";

- `(.*\</quando/>)?`

Permite opcionalidade na marca de fecho global da *<quando>*, pois como vimos, esta pode surgir de formas semelhantes a: *<quando>1965</quando>*, em que queremos extrair o ano.

```
\<quem{espaco}?\">.+\\
```

Este filtro apanha todas as linhas com a marca *<quem>*, e posteriormente será verificado se a marca de fecho existe na mesma linha, caso exista o programa segue o seu fluxo natural, caso contrário **irá entrar num estado que se pode considerar "À caça da marca de fecho", o estado <NOME_NO_END>**. Caso isto aconteça o texto filtrado até ao momento é guardado em memória (neste caso na variável **char* nome**), para posteriormente quando apanharmos a marca de fecho com o filtro:

```
<NOME_NO_END>.*\</quem\>\\
```

sabermos que temos de concatenar o texto já existente em memória com o restante. Já fizemos referência a esta estratégia mas decidimos aqui explicitá-la com mais detalhe pois **é um ponto crítico deste trabalho**.

2.3.1 Condições de contexto

Para além da condição de contexto já apresentada nesta secção, foi implementada outra com objetivos similares **senão o mesmo iguais do ponto de vista funcional**:

- **%x FACTO_NO_END** - permite-nos tratar informação incompleta no caso da marca *facto*. No seguimento da condição de contexto anterior entramos num estado à procura da respetiva marca de fecho (esta poderá não existir pelo que cabe ao filtro "garbage collector" tratar de tais situações).

2.4 Concepção

2.4.1 Estruturas de dados

Numa prespetiva simplista foram implementadas duas *listas ligadas*, uma correspondente à pessoa e uma outra que agrega toda a informação relativa a uma fotografia. A cada elemento da lista ligada de pessoas existe uma lista de ligada de fotos associada, portanto temos um apontador para uma estrutura foto na estrutura pessoa, ou se quisermos, num contexto mais realista, a cada pessoa fazemos corresponder o seu álbum de fotografias.² **Uma estrutura de dados do ponto de vista abstracto por si só não é claro, pelo que é feita uma tentativa de "materializar" a estrutura num esquema.**

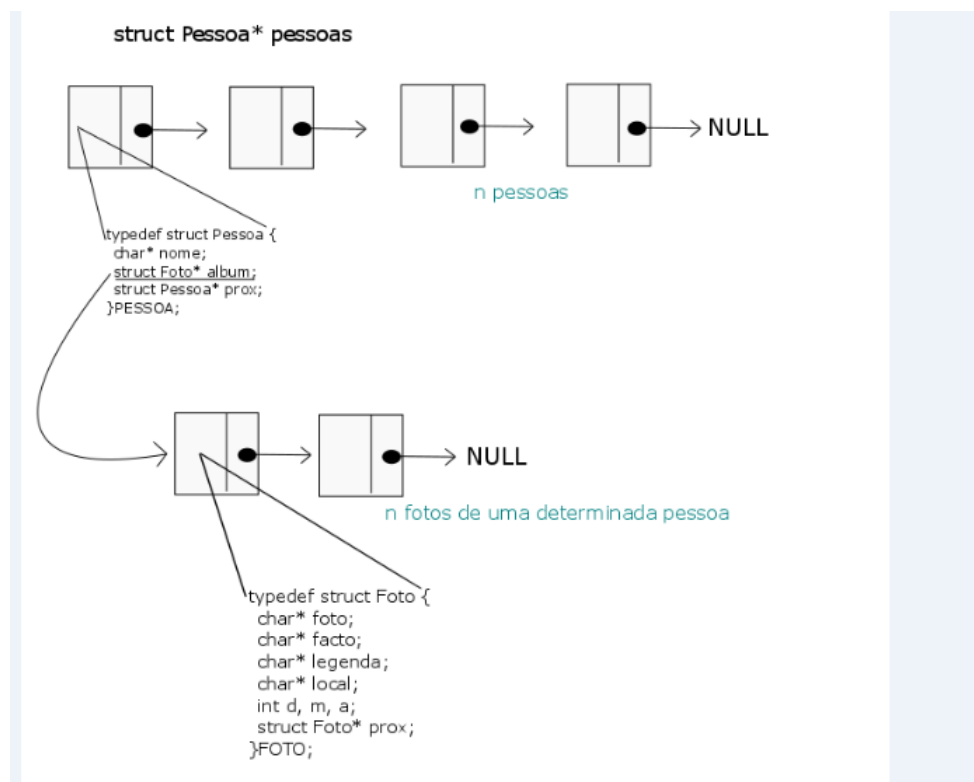


Figura 2.2: Estrutura de dados onde é armazenada a informação resultante de filtrar os diversos ficheiros *.xml*.

²O identificador da pessoa é a marca *<quem>* dos ficheiros *.xml*, portanto para cada marca *<quem>* diferente teremos um elemento na lista ligada de pessoas, esse mesmo elemento por sua vez pode conter N fotos associadas.

2.4.2 Algoritmos

É um **requisito** deste problema (e dos mais importantes tanto do ponto de vista do utilizador como do programador) que num dado álbum, as fotos fiquem ordenadas por ordem cronológica, e as pessoas que são apresentadas na pagina de entrada por ordem alfabética.

Como estamos a trabalhar com estruturas ligadas por apontadores, foi usado o algoritmo **insertion sort** tanto para inserção de fotos como de pessoas. A vantagem em relação a *arrays* será que não é necessário deslocar (*fazer shift*) os elementos para a direita ou para a esquerda pois podemos simplesmente fazer a troca dos apontadores, sendo o tempo de execução destas operações **constantes**. O pior caso nesta situação será a inserção na cauda da lista e terá um custo $O(n)$, pois teremos de percorrer toda a lista ligada para inserir o elemento no final. Esta é uma das desvantagens das listas ligadas, que no nosso caso pode ser *disfarçado* pelo facto de, para criar as páginas *web* termos de percorrer todas as estruturas sem qualquer distinção.

2.4.3 Módulo em C

Para armazenar os dados filtrados foi criado um módulo em C **gerealbum.c** (e respetivo header **gerealbum.h**), com funções que permitem essencialmente **inserir (ordenadamente) elementos na estrutura e gerar ficheiros com format HTML apresentável**. Foram implementadas outras funções auxiliares mas não são relevantes para este ponto.

```
224 <*>.*\</foto>
225     inserePessoa(nome, foto, facto, legenda, local, d, m, a)
226
227     foto=nome=facto=legenda=local=NULL
228     d=-1; m=-1; a=-1
229
```

Figura 2.3: Captura da marca de fecho de uma foto, e inserção dos dados de uma nova foto (consequentemente atualização ou inserção (de nova) pessoa) na estrutura de dados.

Como podemos observar na seguinte imagem é utilizada uma função disponibilizada pelo módulo para a inserção dos dados (**gerealbum.c**). Para gerar as páginas HTML é usada uma estratégia peculiar no que toca a nomear as páginas. Como por vezes os nomes das pessoas (marca *<quem>*) podem conter descrições longas e com espaços em branco para cada álbum HTML é dado um nome que corresponde **ao apontado para a estrutura de dados da pessoa em causa**.

```
sprintf(filename, "%p.html", p);
FILE* file = fopen(filename, "w");
```

Figura 2.4: Estratégia para nomear páginas HTML.

Como podemos observar na fig.4 é usada a primitiva do C `sprintf` para criar o nome da página web, em que `p` é do tipo *struct Pessoa**.

```
235 int yywrap()
236 {
237     //printEstruturas();
238     gerarPaginaDeEntrada();
239     gerarAlbuns();
240     freeMem();
241     free(code);
242     free(currentdir);
243     return 1;
244 }
```

Figura 2.5: Chamada de funções para criação das páginas HTML, internas à função *yywrap*.

2.5 Problemas

Problema:

Um dos problemas de implementação desta solução que traçamos foi a gestão de recursos, isto é, imagens que estão espalhadas pelas diferentes diretorias dos *datasets*. É necessário a quando da criação dinâmica das páginas, inserir corretamente o *path* para cada imagem, sendo estas diferentes de diretoria para diretoria e de computador para computador como podemos gerar esses *file paths* programaticamente?

Solução:

Definir uma macro para o prefixo da referência HTML:

```
#define PREFIX "file:/"
```

Declaração de uma variável global `char* currentdir` onde é armazenado o caminho até à pasta **Exercicio1**.

Posteriormente na função `void gerarPaginaDeEntrada()` é concatenado o *datapath*.

Por fim queremos juntar ao *datapath* o **nome do ficheiro da página web** (vimos anteriormente como o mesmo é gerado) criando assim um link.

```

246 int main(int argc, char* argv[])
247 {
248     if(argc==1) return 0;
249     code = strdup(argv[1]); /*Vai permiti
250     currentdir = strdup(getenv("PWD"));
251     yylex();
252 }
253

```

Figura 2.6: Primitiva do sistema operativo que nos permite obter a diretoria atual `getenv("PWD")`.

```

// Definir path, para criar link, para respetivos albuns
path = (char*) malloc(100*sizeof(char));
path = strcat(path,PREFIX);
path = strcat(path,currentdir);
path = strcat(path,"/"); // Para que no final do path exista / (..../)

```

Figura 2.7: Concatenação do *datapath*.

```

// Percorrer lista ligada para criar indice com nomes
for(p=pessoas; p!=NULL; p=p->prox){
    fprintf(file, "\t\t\t\t\t<p align=\\"center\\" class=\\"scroll\\"><a href=\\"%s%s.html\\">%s</a></p>\n", path, p, p->nome);
}

```

Figura 2.8: Criação de um link para um dado álbum de fotografias.

Problema:

Para cada *dataset* queremos uma página com o título da diretoria no caso da Taberna do Fausto, todos os álbuns associados a esta diretoria estarão numa página cujo o título é **Taberna do Fausto**.

Solução:

É definida uma variável global `char* code` que recebe um parâmetro `argv[1]` (fig.6) passado à função `main`, este parâmetro é que define então o título de cada página.

```

// Pequeno título de início da página
if(strcmp(code,"AM")==0){
    fprintf(file, "\t\t\t\t\t<center><font size=\\"10\\">Ant&ocutenio Machado</font></center>\n");
}
else if(strcmp(code,"TF")==0){
    fprintf(file, "\t\t\t\t\t<center><font size=\\"10\\">Taberna do Fausto</font></center>\n");
}
else if(strcmp(code,"NC")==0){
    fprintf(file, "\t\t\t\t\t<center><font size=\\"10\\">Neca Chamin&eacute;</font></center>\n");
}
else if(strcmp(code,"FG")==0){
    fprintf(file, "\t\t\t\t\t<center><font size=\\"10\\">Fausto Gomes</font></center>\n");
}

```

Figura 2.9: Decisão do título da página.

2.6 Shell Scripting

Com o objetivo de sistematizar os testes e a criação do site foram desenvolvidas duas *shell scripts*, uma primeira **run.sh** que percorre cada *dataset* gerando as diversas páginas HTML e enviado a página de entrada para a diretoria *site* (também esta gerada pelo *script*). Uma segunda **clear.sh** que limpa as diretorias eliminando as páginas HTML geradas.

2.7 criarhome.c

Existe um programa à parte que gera a home page do site Museu da pessoa. O motivo pelo qual se encontra à parte é para que o executável não corra sempre que chamamos o programa principal, desta forma estaríamos a gerar uma *homepage* por cada *dataset*.

2.8 Alternativas

Relativamente à gestão de páginas HTML e criação do site o grupo podia ter-se contentado com a criação de álbuns individuais e de testar cada *dataset* individualmente, mas não o fez pois desta forma o nível de dificuldade do desafio aumentava tornando-o mais interessante.

Uma das soluções que se pode estranhar neste código relativamente aos outros dois trabalhos é a substituição do uso da primitiva **stretch** por ciclos *for* e tratamento *"manual"* da extração de conteúdo das marcas XML.

2.10 Resultados

Nesta secção iremos expor uma série de *screenshots* à medida que navegamos no site **Museu da Pessoa**. O código das primeiras três páginas geradas pode ser consultado em anexo vem como os ficheiros xml input (fg.xml é o ficheiro input).



Figura 2.11: Homepage (**home.html**).

Clicando em **Fausto Gomes** ...



Figura 2.12: Página de entrada para o *dataset* Fausto Gomes (**fausto.html**).

Clicando em **Fausto Ferreira Gomes** ...

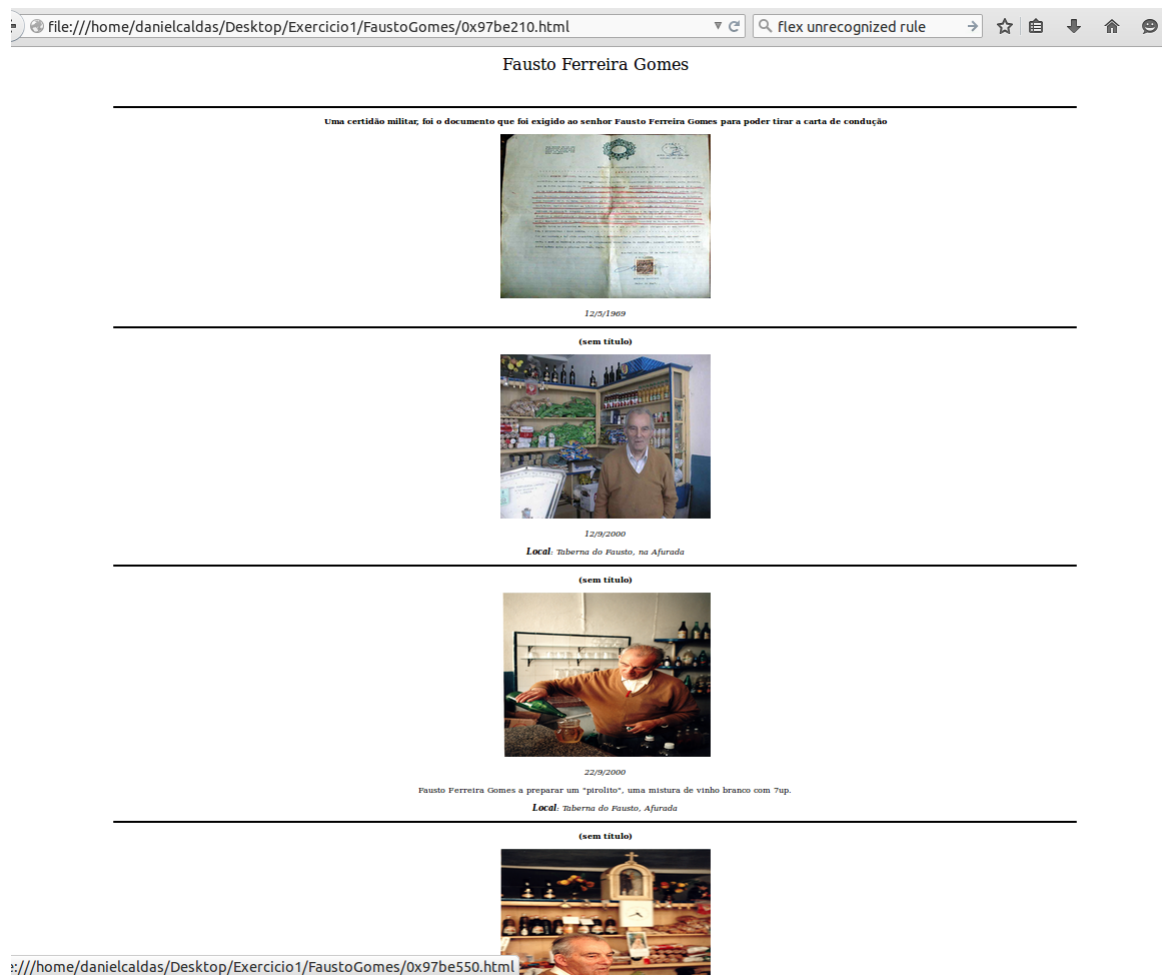


Figura 2.13: Álbum de Fausto Ferreira Gomes (com menos *zoom* de que o habitual para podermos visualizar toda a página) (0x8983210.html).

Outro exemplo já fora do contexto de navegação anterior...



Figura 2.14: Álbum de António Machado.

Capítulo 3

Ficheiros com Canções

3.1 Análise e Especificação

3.1.1 Descrição informal do problema

Para a resolução do problema apresentado, era pedido que desenvolvêssemos um Filtro de Texto com o Flex, que fosse capaz de ler um ficheiro com uma ou mais canções, em que estas estão separada por um linha de hífens iniciada por '##'.

Após a leitura terá de ser criado um documento (da classe article) em LATEX, cujo título seja o indicado no descritor title e o autor seja, ou o indicado no descritor author, ou ambos os indicados como autores da letra e da música.

O ficheiro LATEX terá uma única secção sem numeração e designada por "Letra". Esta secção terá de conter poema devidamente formatado e no fim de tudo, encostado à direita, o cantor, quando conhecido.

3.1.2 Especificação dos Requisitos

Análise dos datasets

Ao analisarmos o ficheiros, disponibilizados concluímos que cada canção é constituída por um cabeçalho, onde se encontra a informação relativa ao título, autor, ..., e pela zona onde se encontra o poema da canção. Quando se encontra uma linha com '##', e um série de hífens, concluímos que se vai proceder à leitura de uma nova canção, tendo então de se guardar os dados relativo á primeira.

Através da análise dos datasets o grupo assumiu que o campo do cabeçalho 'from', aparece sempre em último lugar do cabeçalho, e antes da letra. Em seguida apresentámos um exemplo de um dataset.

```
title: Canta, canta amigo canta  
author: António Macedo
```

```

from: jj

Canta canta amigo canta
vem cantar a nossa canção
tu sozinho não és nada
juntos temos o mundo na mão
...
## _____

```

3.2 Concepção

3.2.1 Estruturas de Dados

Novamente numa prespetiva simplista foi implementada uma lista ligada, referente aos dados de cada canção lida. Em cada elemento da lista ligada é armazenado todos os dados relativos a uma canção, desde o seu título à letra correspondente. **Uma estrutura de dados do ponto de vista abstracto por si só não é claro, pelo que é feita uma tentativa de materializar a estrutura num esquema.**

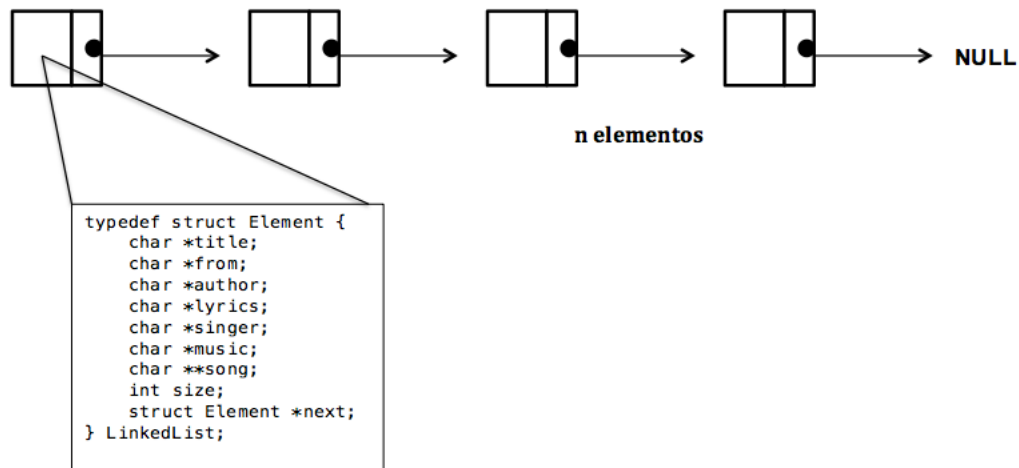


Figura 3.1: Estrutura de dados onde é armazenada a informação resultante de filtrar o ficheiro.

3.2.2 Estratégia

Com base na ideia da secção anterior, o próximo passo do grupo foi delinear uma estratégia para a resolução total do problema. Seguimos então os seguintes passos:

- Escrever em **Latex** um template para os ficheiros a ser criados, por forma a podermos mudar o conteúdo **dinamicamente** através de **funções em C**;
- Análise detalhada dos *datasets* disponibilizados por forma a construirmos um filtro de texto que captasse toda a informação necessária;
- Implementação do filtro de texto em **FLEX**, implementando também algumas funções auxiliares na linguagem C;
- Desenho de uma estrutura de dados que fosse adequada para o problema;
- Implementação de um módulo em C capaz de gerir os dados passados pelo filtro e armazenando-os na estrutura de dados;
- Realização de alguns testes acerca do conteúdo filtrado e de dados armazenados;
- Completar o filtro com funções que geram automaticamente os ficheiros no formato **Latex**;
- Por último criação de **shell scripts** que permitem correr o programa;

3.3 Codificação

3.3.1 Implementação

Para a correta resolução do problema apresentado, foi criado um filtro em FLEX, com o intuito de processar o ficheiro, e armazenar os dados necessários para a criação dos ficheiros LATEX necessários. Foram criadas uma série de expressões regulares, para que, fosse possível apanhar todos os dados relativos a uma canção. Essas expressões regulares são apresentadas de seguida.

espaco	[\t]*	-	faz match dos espaços ou tabs
novaMus	(\#\#)(\-)+	-	faz match do separador de canções
titulo	{espaco}(?i:Title){espaco}:.+	-	faz match com o campo title
from	{espaco}(?i:From){espaco}:.+	-	faz match com o campo from
autor	{espaco}(?i:Author){espaco}:.+	-	faz match com o campo author
musica	{espaco}(?i:Music){espaco}:.+	-	faz match com o campo music
cantor	{espaco}(?i:Singer){espaco}:.+	-	faz match com o campo singer
letra	{espaco}(?i:Lyrics){espaco}:.+	-	faz match com o campo lyrics

Condições de contexto

Através da análise do problema concluímos que quando se processa a leitura do ficheiro, o estado da leitura deve-se alterar mediante certas condições. Com vista a transição entre esses estados de leitura, foram criados 3 três condições de contexto:

- %x HEADER
- %x TEXT
- %x ESP

No estado HEADER, estamos a proceder à leitura do cabeçalho. Quando se termina a leitura do cabeçalho, no nosso caso é quando se lê o campo 'from', a leitura passa para o estado TEXT, ou seja, vai se proceder á leitura do poema da canção, no entanto não guarda as linhas em branco. No estado ESP, também se está a proceder á leitura do poema, mas no entanto, a leitura está a armazenar as linhas em branco entre estrofes.

A transição entre o estado TEXT e o estado ESP é efetuada, logo após se ler uma linha com qualquer caracter excepto o caracter de mudança de linha.

3.3.2 Testes realizados

Para efetuar testes, foi criado um ficheiro, com várias canções, separadas pelo separador já descrito. Para efectuar os testes é necessário que se executem os seguintes passos no terminal:

- make ;
- ./run.sh ;

Após a sua execução vai ser criada uma pasta resultados com as canções divididas em ficheiros com formato pdf. Também podem ser observados os ficheiros em LATEX que correspondem a cada canção.

```
Marcelo ~/Desktop/Exercicio5 - $make
flex filtro.l
gcc lex.yy.c -c
gcc List.c -c -O2 -g -Wall -Wextra
gcc lex.yy.o List.o -o Latex.exe -ll -O2 -g -Wall -Wextra
Marcelo ~/Desktop/Exercicio5 - $./run.sh
This is pdfTeX, Version 3.14159265-2.6-1.40.15 (TeX Live 2014)
```

Figura 3.2: Exemplo do procedimento para testes.

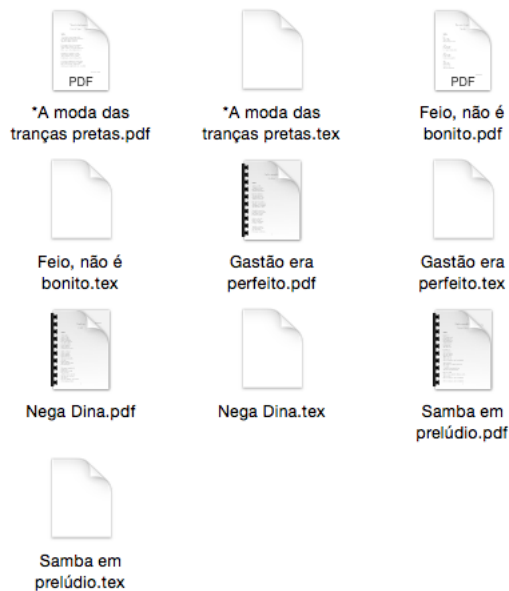


Figura 3.3: Resultados obtidos

3.3.3 Resultados

Nesta subsecção fica exemplificado os resultados obtidos com a realização dos testes efetuados.

Apesar de na Figura 3.3, dar a entender que os ficheiros .tex e .pdf ficam na mesma pasta resultados, isso não é verdade, isso só acontece para melhor exemplificação dos resultados obtidos.

```

\begin{document}

\title{*A moda das tranças pretas}
\author{Vicente da Câmara \and Ginguinhas}

\date{}
\maketitle

\section*{Letra}

Como era linda com seu ar namoradoiro
\Té lhe chamavam "menina das tranças pretas",
\Pelo Chiado passeava o dia inteiro,
\Apregoando raminhos de violetas.
\|E as raparigas de alta roda que passavam
\Ficavam tristes a pensar no seu cabelo,
\Quando ela olhava, com vergonha, disfarçavam
\E pouco a pouco todas deixaram crescê-lo.
\|Passaram dias e as meninas do Chiado
\Usavam tranças enfeitadas com violetas,
\Todas gostavam do seu novo penteado,
\E assim nasceu a moda das tranças pretas.
\|Da violeira já ninguém hoje tem esperanças,
\Deixou saudades, foi-se embora e à tardinha
\Está o Chiado carregado de mil tranças
\Mas tranças pretas ninguém tem como ela as tinha.
\|begin{flushright}
Vicente da Câmara
\end{flushright}

\end{document}

```

Figura 3.4: Ficheiro latex gerado.

*A moda das tranças pretas

Vicente da Câmara Ginguinhas

Letra

Como era linda com seu ar namoradoiro
Té lhe chamavam "menina das tranças pretas",
Pelo Chiado passeava o dia inteiro,
Apregoando raminhos de violetas.

E as raparigas de alta roda que passavam
Ficavam tristes a pensar no seu cabelo,
Quando ela olhava, com vergonha, disfarçavam
E pouco a pouco todas deixaram crescê-lo.

Passaram dias e as meninas do Chiado
Usavam tranças enfeitadas com violetas,
Todas gostavam do seu novo penteado,
E assim nasceu a moda das tranças pretas.

Da violeira já ninguém hoje tem esperanças,
Deixou saudades, foi-se embora e à tardinha
Está o Chiado carregado de mil tranças
Mas tranças pretas ninguém tem como ela as tinha.

Vicente da Câmara

Figura 3.5: Ficheiro pdf gerado.

Capítulo 4

Enamex

4.1 Nota

Este trabalho não foi desenvolvido na totalidade como iremos ver, **foi desenvolvido apenas com o intuito de aprofundar os conhecimentos de manipulação expressões regulares e concepção de filtros de texto utilizando o *FLEX* como ferramenta.**

4.2 Análise e especificação

4.2.1 Descrição informal do problema

Neste exercício é pedido que dum ficheiro XML escrito no dialeto *Enamex* se crie uma página HTML onde se listem cidades, países, pessoas e organizações, no entanto como o grupo já completou na totalidade dois dos trabalhos, neste iremos apenas implementar o filtro base.

4.2.2 Especificação dos requisitos

Análise do ficheiro teste

Ao processar um ficheiro com as características do Enamex podemos falar em essencialmente quatro marcas diferentes que derivam da seguinte frase:

```
<ENAMEX TYPE='XXXXX' SUBTYPE='YYYYY'>NNNNN</ENAMEX>
```

Os casos possíveis considerados são então:

- **XXXXX** - LOCATION, com o *subtype* **YYYYY** variando entre CITY E COUNTRY;
- **XXXXX** - PERSON, uma determinada pessoa;
- **XXXXX** - ORGANIZATION, o nome de uma organização;

4.3 Filtros de texto

Mediante a análise feita na secção anterior foram então desenvolvidos os filtros de texto com as seguintes características:

```
\<ENAMEX\ TYPE\="PERSON"\>.+<\>/ENAMEX\>:
```

Para que se possa filtrar o nome de uma pessoa.

```
cidade \<ENAMEX\ TYPE\="LOCATION\[ \t]*(SUBTYPE\="CITY")?>.+<\>/ENAMEX\>:
```

Para que se possa filtrar o nome de uma cidade sendo o subtipo opcional.

```
pais \<ENAMEX\ TYPE\="LOCATION\[ \t]*(SUBTYPE\="COUNTRY")?>.+<\>/ENAMEX\>:
```

Igual ao anterior, mas desta vez para países.

```
org \<ENAMEX\ TYPE\="ORGANIZATION"\>.+<\>/ENAMEX\>:
```

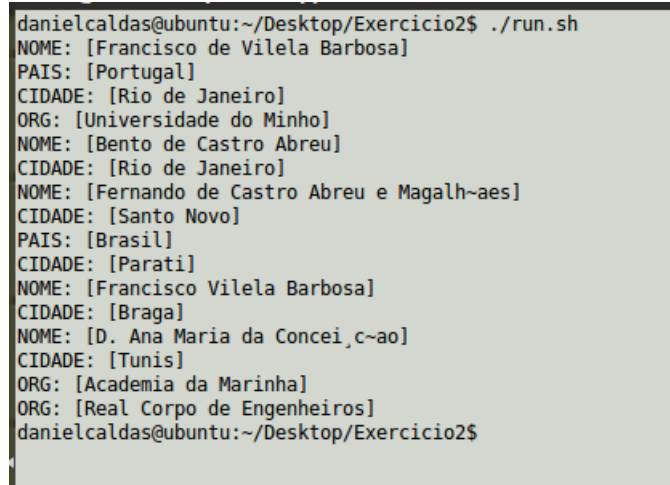
Para filtrarmos o nome de organização.

4.4 Extrair conteúdo

Como o conteúdo a extrair das diferentes marcas é estruturalmente o mesmo (embora diferente em termos de semântica), foi definida a função *char* extract_value* que retira as marcas de início e fim.

4.5 Testes e Resultados

Foi criado um ficheiro texto a partir do disponibilizado no enunciado para podermos verificar a nossa solução.



```
danielcaldas@ubuntu:~/Desktop/Exercicio2$ ./run.sh
NOME: [Francisco de Vilela Barbosa]
PAIS: [Portugal]
CIDADE: [Rio de Janeiro]
ORG: [Universidade do Minho]
NOME: [Bento de Castro Abreu]
CIDADE: [Rio de Janeiro]
NOME: [Fernando de Castro Abreu e Magalhães]
CIDADE: [Santo Novo]
PAIS: [Brasil]
CIDADE: [Parati]
NOME: [Francisco Vilela Barbosa]
CIDADE: [Braga]
NOME: [D. Ana Maria da Conceição]
CIDADE: [Tunis]
ORG: [Academia da Marinha]
ORG: [Real Corpo de Engenheiros]
danielcaldas@ubuntu:~/Desktop/Exercicio2$
```

Figura 4.1: Output do processamento do ficheiro teste.

Na fig. 3.3 podemos observar como o conteúdo das entidades *Enamex* é extraído corretamente.

Capítulo 5

Conclusão geral

5.1 Aprendizagem

Terminada esta série de trabalhos práticos podemos concluir que atingimos os pontos listados em baixo:

- Análise de detalhada de meta-dados e construção de ficheiros, em diversos formatos desde o texto, passando pelo LaTeX, HTML até ao XML e **seus dialetos**;
- Criação de filtros de texto em **FLEX** capazes de extraír a informação com base em requisitos pré-especificados;
- Uso de condições de contexto para criação de filtros para **ficheiros mal formados**;
- Consolidação dos conhecimentos de manipulação de estruturas de dados e algoritmos;
- Consolidação dos conhecimentos em linguagem C e ferramentas Unix;
- Estudo de alguns conceitos de *Shell Scripting* e *Bash*;

5.2 Análise Crítica geral

Estamos muito satisfeitos com o trabalho desenvolvido no entanto temos noção que se o grupo tivesse sido mais objetivo teria resolvido mais trabalhos práticos, no entanto explorar com mais detalhe os problemas pode ser igualmente (e foi) satisfatório.

Apêndice A

Código do Programa 2.1

A.0.1 filtro.1

```
1 foto [0-9]{3}\-[A-Z]\-[0-9]{2}\.((?i:png)|(?i:jpg)|(?i:gif))
2 espaco [ \t]
3 data [0-9]+(\-|\.)[0-9]+(\-|\.)[0-9]+
4
5 %{
6     #include <ctype.h>
7     #include "gerealbum.h"
8
9     char* foto=NULL;
10    char* nome=NULL;
11    char* facto=NULL;
12    char* legenda=NULL;
13    char* local=NULL;
14    char param[5];
15    int d, m, a;
16
17
18    // Algumas funções para o tratamento de
19    strings
20
21    /**
22     *Função que retira espaços do início e fim de uma
23     string.
24     *@param recebe uma string.
25     *@return retorna apontador para string modificada.
26     */
27    char* trim(char* token)
28    {
29        int i, n;
```

```

30         ;
31         token+=i;
32         n = strlen(token);
33         for (i=n-1; token[i]!=' ' || token[i]!='\t'; i
            --)
34         ;
35         token[i+1]='\0';
36
37         return token;
38     }
39
40     /**
41     * Função auxiliar para remoção de tags.
42     *@param string com elementos a retirar.
43     *@return apontador para nova string sem as tags.
44     */
45     char* extract_attribute(char* token)
46     {
47         int i;
48
49         for (i=0; token[i]!='"' && token[i]!='\0'; i++)
50         ;
51         i++;
52         token+=i;
53         for (i=strlen(token); token[i]!='"' && i>=0; i
            --)
54         ;
55         token[i]='\0';
56
57         return strdup(token);
58     }
59
60     /**
61     * Função que verifica se frase em xml termina
        corretamente.
62     *@param frase em xml.
63     *@param marca de fecho dessa frase.
64     *@return 1 caso frase termine com tag de fecho, 0
        caso contrário.
65     */
66     int check_end(char* line, char* mark)
67     {
68         int flag_match=0;
69         int i, j;
70
71         for (i=0; line[i]!='\0' && line[i]!='\n';) {
72             if (line[i]==mark[0]) {
73                 flag_match=1;
74                 for (j=0; mark[j]!='\0' && mark
                     [j]!='\n' && line[i]!='\0'

```



```

75         && line[i]!='\n'; j++, i
        ++){
            if (line[i]!=mark[j]){
                flag_match=0;
                break;}
76     }
77     if (flag_match==1) break;
78 }
79 else {
80     i++;
81 }
82 }
83
84     return flag_match;
85 }
86 %}
87
88 %x FACTO_NO_END
89 %x NOME_NO_END
90
91 %%
92
93 .*\

```

```

118         }
119         param[k]='\\0';
120         k = atoi(param);
121         switch(j) {
122             case 0:
123                 a=k;
124                 break;
125             case 1:
126                 m=k;
127                 break;
128             case 2:
129                 d=k;
130                 break;
131         }
132         i++;
133     }
134 }
135 }
136
137 \\<quem{espaco}?\\>.+ {
138     int i;
139
140     for (i=0; yytext[i]!='>'; i++)
141     ;
142     i++;
143     yytext+=i;
144
145     if (check_end(yytext, "</quem>")==1){
146         for (i=strlen(yytext)-1; yytext[i]!='<' &&
147             yytext[i+1]!='/' && yytext[i+2]!='q'; i--)
148         ;
149         yytext[i]='\\0';
150
151         nome = strdup(yytext);
152         nome = trim(nome);
153     }
154     else {
155         nome = strdup(yytext);
156         BEGIN(NOME_NO_END);
157     }
158 }
159
160 <NOME_NO_END>.*\\<\\quem\\> {
161     int i;
162     nome = trim(nome);
163     yytext = trim(yytext);
164     // Bloco para realocar e concatenar nome
165     unsigned int size = strlen(nome)+strlen(yytext)+2;
166     char* aux = strdup(nome);

```

```

167     nome = (char*) malloc(size*sizeof(char));
168     nome = strcat(nome,aux);
169     nome = strcat(nome," ");
170     nome = strcat(nome,yytext);
171
172     for (i=strlen(nome)-1; nome[i]!='<' && nome[i+1]!='/'
        && nome[i+1]!='q'; i--)
173     ;
174     nome[i-1]='\0';
175
176     BEGIN(INITIAL);
177 }
178
179 \<facto{espaco}?>\.+ {
180     int i;
181
182     for (i=0; yytext[i]!='\n' && yytext[i]!='\0' && yytext[
        i]!='>'; i++)
183     ;
184     i++;
185     yytext+=i;
186
187     if (check_end(yytext, "</facto>")==1){
188         for (i=strlen(yytext); yytext[i]!='<'; i--)
189         ;
190         yytext[i]='\0';
191         facto = strdup(yytext);
192     }
193     else {
194         facto = strdup(yytext);
195         BEGIN(FACTO_NO_END);
196     }
197 }
198
199 <FACTO_NO_END>.*\<\/facto\> {
200     int i;
201
202     facto = trim(facto);
203     yytext = trim(yytext);
204
205     // Bloco para realocar e concatenar facto
206     unsigned int size = strlen(facto)+strlen(yytext)+2;
207     char* aux = strdup(facto);
208     facto = (char*) malloc(size*sizeof(char));
209     facto = strcat(facto,aux);
210     facto = strcat(facto," ");
211     facto=strcmp(facto,yytext);
212
213     for (i=strlen(facto)-1; facto[i]!='<'; i--)
214     ;

```

```

215         fact o [ i ] = '\0';
216
217         BEGIN( INITIAL );
218     }
219
220     \<legenda{espaco}? \> . + \< \ / legenda \> {
221         int i;
222
223         for ( i = 0; yytext [ i ] != '\n' && yytext [ i ] != '\0' && yytext [
                i ] != '>'; i++)
224             ;
225         i++;
226         yytext += i;
227
228         for ( i = strlen ( yytext ); yytext [ i ] != '<'; i--)
229             ;
230         yytext [ i ] = '\0';
231         legenda = strdup ( yytext );
232         legenda = trim ( legenda );
233     }
234
235     \<onde{espaco}? \> . + \< \ / onde \> {
236         int i;
237
238         for ( i = 0; yytext [ i ] != '\n' && yytext [ i ] != '\0' && yytext [
                i ] != '>'; i++)
239             ;
240         i++;
241         yytext += i;
242
243         for ( i = strlen ( yytext ); yytext [ i ] != '<'; i--)
244             ;
245         yytext [ i ] = '\0';
246         local = strdup ( yytext );
247         local = trim ( local );
248     }
249
250
251     <* > . * \< \ / foto > {
252         inserePessoa ( nome , foto , fact o , legenda , local , d , m , a );
253         foto = nome = fact o = legenda = local = NULL;
254         d = -1; m = -1; a = -1;
255     }
256
257
258     <* > . | \n { ; }
259
260     %%
261
262     int yywrap ()

```

```
263 {
264     // printEstruturas();
265     gerarPaginaDeEntrada();
266     gerarAlbuns();
267     freeMem();
268     free(code);
269     free(currentdir);
270     return 1;
271 }
272
273 int main(int argc, char* argv[])
274 {
275     if(argc==1) return 0;
276     code = strdup(argv[1]); /*Vai permitir-nos colocar o
        path certo para as diversas imagem do Site Museu
        da Pessoa*/
277     currentdir = strdup(getenv("PWD"));
278     yylex();
279 }
```

A.0.2 gerealbum.c

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include <ctype.h>
5
6 #include "gerealbum.h"
7
8 #define PREFIX "file://"
9
10 // Estruturas de dados do módulo
11 typedef struct Foto {
12     char* foto;
13     char* facto;
14     char* legenda;
15     char* local;
16     int d, m, a;
17     struct Foto* prox;
18 }FOTO;
19
20 typedef struct Pessoa {
21     char* nome;
22     struct Foto* album; // Coleção de fotos de uma
        determinada pessoa
23     struct Pessoa* prox;
24 }PESSOA;
25
```

```

26 // Vari vel global para armazenamento de pessoas e respectivos
    albuns
27 struct Pessoa* pessoas;
28
29
30 /**
31  *Comparar duas data cronologicamente.
32  *@param FOTO* f1, apontador para primeira estrutura foto.
33  *@param FOTO* f2, apontador para estrutura correspondente
    segunda foto.
34  *@return int, 0 caso datas sejam iguais, 1 caso f1>f2, -1
    caso contr rio.
35  */
36 int comparaDatas(struct Foto* f1, struct Foto* f2)
37 {
38     if(f1->a > f2->a) return 1;
39     if(f1->a == f2->a){
40         if(f1->m > f2->m) return 1;
41         else if(f1->m == f2->m){
42             if(f1->d > f2->d) return 1;
43             else if(f1->d == f2->d) return 0;
44         }
45     }
46     return -1;
47 }
48
49
50 /**
51  *Inser o de fotos num alb m de uma dada pessoa.
52  *@param album, apontador para o lbum de fotos da pessoa.
53  *@param foto, nome da nova foto a inserir.
54  *@param facto, o t tulo da foto.
55  *@param legenda, poss vel legenda da foto. (parametro EXTRA)
56  *@param local, local onde foi tirada a foto. (parametro EXTRA
    )
57  *@param d, m, a, data da foto.
58  */
59 struct Foto* insereFoto(struct Foto* album, char* foto, char*
    facto, char* legenda, char* local, int d, int m, int a)
60 {
61     if(album==NULL){
62         album = (FOTO*) malloc(sizeof(struct Foto));
63         album->foto=strdup(foto);
64         if(facto!=NULL){
65             album->facto = (char*) malloc(strlen(
                facto)*sizeof(char));
66             strcpy(album->facto, facto);
67         }
68         else album->facto="(sem t&iacutetitulo)";

```

```

69         if (legenda!=NULL) album->legenda=strdup(
70             legenda);
71         if (local!=NULL) album->local=strdup(local);
72         album->d=d;
73         album->m=m;
74         album->a=a;
75         album->prox=NULL;
76         return album;
77     }
78     struct Foto* nova = (FOTO*) malloc(sizeof(struct Foto)
79         );
80     // Preencher nova estrutura
81     nova->foto=strdup(foto);
82     if (facto!=NULL) nova->facto=strdup(facto);
83     else nova->facto="(sem t&iacutetitulo) ";
84     if (legenda!=NULL) nova->legenda=strdup(legenda);
85     if (local!=NULL) nova->local=strdup(local);
86     nova->d=d;
87     nova->m=m;
88     nova->a=a;
89     nova->prox=NULL;
90
91     int r;
92     int flag=0;
93     struct Foto* x;
94     struct Foto* ant;
95     struct Foto* aux;
96
97     for (ant=album, x=album; x!=NULL && flag!=1; ant=x, x=x
98         ->prox) {
99         r=comparaDatas(x, nova);
100         switch(r) {
101             case 0: // As datas s o iguais
102                 aux=x->prox;
103                 x->prox=nova;
104                 nova->prox=aux;
105                 flag=1;
106                 break;
107             case -1: // Data de nova posterior
108                 data de x
109                 if (x->prox==NULL) {
110                     x->prox=nova;
111                     nova->prox=NULL;
112                     flag=1;
113                 }
114                 break;
115             case 1: // Data de nova anterior
116                 data de x

```

```

114         if (ant!=x){
115             ant->prox=nova;
116             nova->prox=x;
117         } else {
118             album=nova;
119             nova->prox=x; //
                Inser o
                cabe a da lista
120         }
121         flag=1;
122         break;
123     }
124 }
125
126     return album;
127 }
128
129 /**
130  *Inserir uma pessoa na estrutura de dados com atualiza o
    ou cria o do respectivo lbum de fotos.
131  *@param nome, o nome da pessoa-
132  *@param foto, nome do ficheiro/foto.
133  *@param facto, t tulo da foto.
134  *@param legenda, uma legenda da fotografia. (parametro EXTRA)
135  *@param local, local onde foi tirada a fota o (parametro
    EXTRA)
136  *@param d, m, a, s o a data da foto, dia, m s e ano (
    respetivamente).
137  */
138 void inserePessoa(char* nome, char* foto, char* facto, char*
    legenda, char* local, int d, int m, int a)
139 {
140     // S s o aceites fotos com a marca <quem>
141     if(nome!=NULL){
142         // Caso inicial, lbum de pessoas vazio
143         if(pessoas==NULL){
144             pessoas = (PESSOA*) malloc(sizeof(
                struct Pessoa));
145             pessoas->nome = strdup(nome);
146             pessoas->album=NULL;
147             pessoas->prox=NULL;
148
149             pessoas->album = insereFoto(pessoas->
                album,foto,facto,legenda,local,d,m
                ,a);
150         }
151         else { // Inser o de uma nova pessoa OU
            atualiza o do lbum de uma j
            existente
152             struct Pessoa* pnova;

```



```

153     struct Pessoa* atual;
154     struct Pessoa* ant;
155     int r;
156     int flag=0;
157     for (atual=pessoas, ant=pessoas; atual
        !=NULL && flag!=1; atual=atual->
        prox) {
158         r=strcmp(nome, atual->nome);
159         if (r==0){ // Atualizar lbum
            de uma pessoa
160             atual->album =
                insereFoto(atual->
                album, foto, facto,
                legenda, local, d, m,
                a);
161             flag=1;
162         }
163         else if (r<0){ // Inserir o
            esquerda
164             pnova = (PESSOA*)
                malloc(sizeof(
                struct Pessoa));
165             pnova->nome = strdup(
                nome);
166             pnova->album=NULL;
167             pnova->prox=NULL;
168             if (ant!=atual){
169                 ant->prox=
                    pnova;
170                 pnova->prox=
                    atual;
171             } else {
172                 pessoas=pnova;
                    //
                    Inserir o
                    cabe a da
                    lista
173                 pessoas->prox=
                    atual;
174             }
175             pnova->album =
                insereFoto(pnova->
                album, foto, facto,
                legenda, local, d, m,
                a);
176             flag=1;
177         }
178         else if (atual->prox==NULL){ //
            Proximo elemento nulo,

```

```

179         inser o na cauda (Caso
180         contr rio continuar a
181         procurar)
182         pnova = (PESSOA*)
183             malloc(sizeof(
184                 struct Pessoa));
185         pnova->nome = strdup(
186             nome);
187         pnova->album=NULL;
188         pnova->prox = NULL;
189         pnova->album =
190             insereFoto(pnova->
191                 album, foto, facto,
192                 legenda, local, d, m,
193                 a);
194         atual->prox=pnova;
195         flag=1;
196     }
197 }
198 }
199 }
200 }
201
202 /**
203  * Fun o que liberta mem ria ocupada pela lista ligada .
204  */
205 void freeMem()
206 {
207     struct Foto* faux;
208     struct Foto* f;
209     struct Pessoa* paux;
210     struct Pessoa* p;
211
212     for(p=pessoas; p->prox!=NULL; p=paux){
213         for(f=p->album; f->prox!=NULL; f=faux){
214             faux=f->prox;
215             free(f);
216         }
217         free(f);
218         paux=p->prox;
219         free(p);
220     }
221
222     // Libertar ltima estrutura que aponta para NULL
223     for(f=p->album; f->prox!=NULL; f=faux){
224         faux=f->prox;
225         free(f);
226     }
227     free(f);
228     free(p);

```

```

219 }
220
221 /**
222  *Gerar p gina de acesso aos albuns, cujo respetivo link fica
223   na home page do site Museu da Pessoa.
224 */
225 void gerarPaginaDeEntrada()
226 {
227     struct Pessoa* p;
228     FILE* file = fopen("home.html", "w");
229     char* path;
230
231     // Definir path, para criar link, para respetivos
232     albuns
233     path = (char*) malloc(100*sizeof(char));
234     path = strcat(path,PREFIX);
235     path = strcat(path,currentdir);
236     path = strcat(path,"/"); // Para que no final do path
237     exista / (.../)
238
239     fprintf(file, "<!DOCTYPE html>\n");
240     fprintf(file, "<head>\n");
241     fprintf(file, "\t<meta char-set=\"utf8\"/>\n");
242     fprintf(file, "<title>Museu da Pessoa</title>\n");
243
244     fprintf(file, "\t<style type=\"text/css\">\n");
245     fprintf(file, "\t\t#wrapper{\n");
246     fprintf(file, "\t\t\ttext-align:center;\n");
247     fprintf(file, "\t\t\tmargin-top:0px;\n");
248     fprintf(file, "\t\t\tmargin-bottom:0px;\n");
249     fprintf(file, "\t\t\tpadding:0px;\n\t}");
250     fprintf(file, "\t</style>\n");
251
252     fprintf(file, "</head>\n");
253     fprintf(file, "<body>\n");
254     fprintf(file, "\t<div id=\"wrapper\">\n");
255     fprintf(file, "\t\t\t<nav>\n");
256
257     // Pequeno t tulo de in cio da p gina
258     if(strcmp(code,"AM")==0){
259         fprintf(file, "\t\t\t\t<center><font size
260         =\"10\">Ant&ocutenio Machado</font></
261         center>\n");
262     }
263     else if(strcmp(code,"TF")==0){
264         fprintf(file, "\t\t\t\t<center><font size
265         =\"10\">Taberna do Fausto</font></center>\n");
266     }
267     else if(strcmp(code,"NC")==0){

```

```

262         fprintf(file , "\t\t\t<center><font size
           =\10\>Neca Chamin&eacute</font></center>
           >\n");
263     }
264     else if (strcmp(code,"FG")==0){
265         fprintf(file , "\t\t\t<center><font size
           =\10\>Fausto Gomes</font></center>>\n");
266     }
267
268
269     fprintf(file , "\t\t\t<center><font size=\6\>&
           Iacutendice</font></center>>\n");
270
271     fprintf(file , "\t\t\t<ul>\n");
272
273     // Percorrer lista ligada para criar ndice com nomes
274     for(p=pessoas; p!=NULL; p=p->prox){
275         fprintf(file , "\t\t\t\t<p align=\"center\"
           class=\"scroll\"><a href=\"%s%p.html\">%s
           </a></p>\n", path, p, p->nome);
276     }
277     fprintf(file , "\t\t\t</ul>");
278     fprintf(file , "\t\t</nav>\n\t\t<br>\n");
279     fprintf(file , "\t</div>\n");
280     fprintf(file , "</body>\n</html>\n");
281 }
282
283 /**
284  *Gerar para uma dada pessoa o seu lbum HTML.
285  *@param struct Pessoa* p, apontador para estrutura com dados
           e lbum de fotos da pessoa.
286  */
287 void gerarPaginaPessoal(struct Pessoa* p)
288 {
289     struct Foto* x;
290
291     // Nome da p gina html
292     char* filename = malloc((strlen(p->nome))*sizeof(char)
           );
293     sprintf(filename,"%p.html",p);
294
295     FILE* file = fopen(filename , "w");
296
297     fprintf(file , "<!DOCTYPE html>\n");
298     fprintf(file , "<head>\n");
299     fprintf(file , "\t<meta char-set=\"utf8\"/>\n");
300     fprintf(file , "<title>Museu da Pessoa - %s</title>\n",
           p->nome); // Dar t tulo apropriado p gina
301
302     fprintf(file , "\t<style type=\"text/css\">\n");

```

```

303     fprintf(file, "\t#wrapper{\n");
304     fprintf(file, "\t\ttext-align:center;\n");
305     fprintf(file, "\t\tmargin-top:0px;\n");
306     fprintf(file, "\t\tmargin-bottom:0px;\n");
307     fprintf(file, "\t\tpadding:0px;\n\t}");
308     fprintf(file, "\t</style>\n");
309
310     fprintf(file, "</head>\n");
311     fprintf(file, "<body>\n");
312     fprintf(file, "\t<div id=\"wrapper\">\n");
313
314     fprintf(file, "\t\t\t<p align=\"center\"><font size
        =6>%s</font></p>", p->nome);
315     fprintf(file, "\t\t\t<br><br>\n");
316     fprintf(file, "\t\t\t<hr color=\"black\" size=\"3\"
        width=\"80%%\">\n");
317
318
319     //Percorrer lista ligada para criar secões com as
        fotos e respetiva informação
320     for(x=p->album; x!=NULL; x=x->prox){
321         fprintf(file, "\t\t<div id=\"%s\">\n", x->foto);
322         fprintf(file, "\t\t\t<ul>\n");
323         fprintf(file, "\t\t\t\t<p><b>%s</b></p>\n", x->
            facto);
324         fprintf(file, "\t\t\t\t<img src=\"%s\" style=\"
            width:420px;height:328px\">\n", x->foto);
325         if(x->a!=-1 && x->m!=-1 && x->d!=-1){
326             fprintf(file, "\t\t\t\t\t<p><i>%d/%d/%d</i></p>\n",
                x->d, x->m, x->a);
327         }
328         else if(x->a!=-1){
329             fprintf(file, "\t\t\t\t\t<p><i>%d</i></p>\n", x
                ->a);
330         }
331         else{
332             fprintf(file, "\t\t\t\t\t<p><i>(sem data)</i></p>
                >\n");
333         }
334         if(x->legenda!=NULL){
335             fprintf(file, "\t\t\t\t\t<p>%s</p>\n", x->
                legenda);
336         }
337         if(x->local!=NULL){
338             fprintf(file, "\t\t\t\t\t<p><i><b>Local</b>: %s
                </i></p>\n", x->local);
339         }
340         fprintf(file, "\t\t\t\t</ul>\n");
341         fprintf(file, "\t\t\t<hr color=\"black\" size=\"3\"
            width=\"80%%\">\n");

```

```

342         fprintf(file , "\t\t</div>\n");
343     }
344
345     fprintf(file , "\t\t</div>\n");
346     fprintf(file , "</body>\n");
347     fprintf(file , "</html>\n");
348
349     fclose(file);
350 }
351
352 /**
353  *Gerar lbuns html individuais para cada pessoa.
354  */
355 void gerarAlbuns()
356 {
357     struct Pessoa* p;
358     // Criar uma pagina html para cada pessoa diferente
359     // com o respectivo lbun
360     for(p=pessoas; p!=NULL; p=p->prox){
361         gerarPaginaPessoal(p);
362     }
363
364 /**
365  *Pequena funcao que imprime as estruturas de dados para
366  *stdout, para efeitos de teste.
367  */
368 void printEstruturas()
369 {
370     struct Foto* f;
371     struct Pessoa* p;
372
373     for(p=pessoas; p!=NULL; p=p->prox){
374         printf("-----%s-----\n", p->nome);
375         for(f=p->album; f!=NULL; f=f->prox){
376             printf("FOTO: [%s]\n", f->foto);
377             if(f->facto!=NULL) printf("%s\n", f->facto);
378             ;
379             printf("%d/%d/%d\n", f->d, f->m, f->a);
380             if(f->legenda!=NULL) printf("%s\n", f->
381                 legenda);
382             if(f->local!=NULL) printf("%s\n", f->local);
383             ;
384         }
385         printf("-----\n");
386     }
387 }

```

A.0.3 gerealbum.h

```
1 #ifndef _GEREALBUM_H
2 #define _GEREALBUM_H
3
4 char* code;
5 char* currentdir;
6
7 // Funções visíveis para fora do módulo
8 void inserePessoa(char* nome, char* foto, char* facta, char*
    legenda, char* local, int d, int m, int a);
9 void freeMem();
10 void gerarPaginaDeEntrada();
11 void gerarAlbuns();
12 void printEstruturas();
13
14 #endif
```

A.0.4 makefile

```
1 FLAGS= -O2 -g -Wall -Wextra
2
3 main: pessoa.out
4
5 pessoa.out: lex.yy.o gerealbum.o
6         gcc lex.yy.o gerealbum.o -o pessoa.exe -ll
           (FLAGS)lex.yy.o: lex.yy.c gcclex.yy.c - clex.yy.c: filtro.lgerealbum.hgerealbum.cflexfiltro.lgerealbum.c
           (FLAGS)
7
8 clean:
9         -rm *.o
10        -rm lex.yy.c
11        -rm *.exe
```

A.0.5 criarhome.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5
6 #define PREFIX "file://"
7 #define SITE_FOLDER "/site/"
8
9 int main()
10 {
```

```

11     char* currentdir;
12     currentdir = strdup(getenv("PWD"));
13
14     FILE* file = fopen("home.html", "w");
15     char* path;
16
17     // Definir path para diretoria "site"
18     path = (char*) malloc(100*sizeof(char));
19     path = strcat(path, PREFIX);
20     path = strcat(path, currentdir);
21     path = strcat(path, SITE_FOLDER);
22
23     fprintf(file, "<!DOCTYPE html>\n");
24     fprintf(file, "<head>\n");
25     fprintf(file, "\t<meta char-set=\"utf8\"/>\n");
26     fprintf(file, "<title>Museu da Pessoa</title>\n");
27
28     fprintf(file, "\t<style type=\"text/css\">\n");
29     fprintf(file, "\t#wrapper{\n");
30     fprintf(file, "\t\ttext-align:center;\n");
31     fprintf(file, "\t\tmargin-top:0px;\n");
32     fprintf(file, "\t\tmargin-bottom:0px;\n");
33     fprintf(file, "\t\tpadding:0px;\n\t}");
34     fprintf(file, "\t</style>\n");
35
36     fprintf(file, "</head>\n");
37     fprintf(file, "<body>\n");
38     fprintf(file, "\t<div id=\"wrapper\">\n");
39     fprintf(file, "\t\t<nav>\n");
40
41     fprintf(file, "\t\t\t<center><font size=\"7\">&
        Iacutendice</font></center>\n");
42
43     fprintf(file, "\t\t\t<ul>\n");
44     fprintf(file, "\t\t\t\t<p align=\"center\" class=\"
        scroll\"><a href=\"%santonio.html\">Ant&ocutenio
        Machado</a></p>\n", path);
45     fprintf(file, "\t\t\t\t<p align=\"center\" class=\"
        scroll\"><a href=\"%sfausto.html\">Fausto Gomes</
        a></p>\n", path);
46     fprintf(file, "\t\t\t\t<p align=\"center\" class=\"
        scroll\"><a href=\"%staberna.html\">Taberna do
        Fausto</a></p>\n", path);
47     fprintf(file, "\t\t\t\t<p align=\"center\" class=\"
        scroll\"><a href=\"%sneca.html\">Neca Chamin&
        eacute</a></p>\n", path);
48
49     fprintf(file, "\t\t\t</ul>");
50     fprintf(file, "\t\t</nav>\n\t\t<br>\n");
51     fprintf(file, "\t\t<footer>\n");

```



```
52     fprintf(file , "\t\t\t<p><b>Grupo de trabalho</b></p>\n\n");
53     fprintf(file , "\t\t\t<p><i>Daniel Caldas a67691 /
        Marcelo Gon&ccedilalves a67736 / Ricardo Silva
        a67728</i></p>\n");
54     fprintf(file , "\t\t</footer>\n");
55     fprintf(file , "\t</div>\n");
56     fprintf(file , "</body>\n</html>\n");
57
58     fclose(file);
59
60     return 0;
61 }
```

A.0.6 run.sh

```
1  #!/bin/bash
2  cp pessoa.exe AntonioMachado/
3  cp pessoa.exe TabernaDoFausto/
4  cp pessoa.exe NecaChamine/
5  cp pessoa.exe FaustoGomes/
6
7  mkdir site
8
9  cd AntonioMachado/
10 cat legenda.xml | ./pessoa.exe AM
11 rm pessoa.exe
12 mv home.html antonio.html
13 mv antonio.html ~/Desktop/Exercicio1/site
14 cd ..
15
16 cd TabernaDoFausto/
17 cat legenda.xml | ./pessoa.exe TF
18 rm pessoa.exe
19 mv home.html taberna.html
20 mv taberna.html ~/Desktop/Exercicio1/site
21 cd ..
22
23 cd NecaChamine/
24 cat legenda.xml | ./pessoa.exe NC
25 rm pessoa.exe
26 mv home.html neca.html
27 mv neca.html ~/Desktop/Exercicio1/site
28 cd ..
29
30 cd FaustoGomes/
31 cat legenda.xml | ./pessoa.exe FG
32 rm pessoa.exe
```

```
33 mv home.html fausto.html
34 mv fausto.html ~/Desktop/Exercicio1/site
```

A.0.7 clear.sh

```
1 #!/bin/bash
2 rm home.html
3 rm a.out
4
5 cd AntonioMachado/
6 rm *.html
7 cd ..
8
9 cd TabernaDoFausto/
10 rm *.html
11 cd ..
12
13 cd NecaChamine/
14 rm *.html
15 cd ..
16
17 cd FaustoGomes/
18 rm *.html
19 cd ..
20
21 rm -rf site
```

A.0.8 Ficheiros Input

am.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE fotos SYSTEM "Users/sony/mp/fotos.dtd">
3 <fotos>
4   <foto ficheiro="022-F-01.jpg">
5     <quando data="1961-01-15"/>
6     <quem>Ana de Lourdes de Oliveira Chamine e Antonio
7       Oliveira
8       Machado</quem>
9     <facto> Os noivos cortam o bolo de casamento</facto>
10   </foto>
11   <foto ficheiro="022-F-02.jpg">
12     <onde> Casa Machado, na Afurada</onde>
13     <onde>Casa Machado, Afurada, Vila Nova de Gaia</onde>
14     <quando data="2000-09-12"/>
15     <quem>Ana de Lourdes de Oliveira Chamine e Antonio
16       Oliveira
```

```

15         Machado</quem>
16     <facto>Antonio Machado e a sua esposa , dona Ana atras
        do
17         balcao da taberna Casa Machado.</facto>
18 </foto>
19 <foto ficheiro="022-F-04.jpg">
20     <quando data="1961-01-15"/>
21     <quem>Da esquerda para a direita: Henrique Oliveira
        Machado, irmao de
22         Antonio Machado, Ana de Lourdes, Antonio
        Oliveira Machado e Joaquim
23         Carvalho Machado, pai de Antonio Machado</quem>
        >
24     <facto>Casamento de Antonio Oliveira com Ana de
        Lourdes de Oliveira
25         Chamine.</facto>
26 </foto>
27 <foto ficheiro="022-F-03.jpg">
28     <onde> Colegio Nossa Senhora de Lourdes, Porto</onde>
29     <quando data="1969-06-05"/>
30     <quem> Da esquerda para a direira: Ana de Lourdes
        Oliveira
31         Chamine,Rosa de Castro Oliveira , sogra do
        senhor Machado, Anibal Alves, o
32         sogro, Antonio Oliveira Machado e Eugenia
        Maria de Oliveira
33         Chamine Machado, filha mais velha do casal Ana
        Chamine e Antonio
34         Machado</quem>
35     <facto>Primeira comunhao de Eugenia Maria de Oliveira
        Chamine Machado, filha mais velha de Ana
36         Chamine e Antonio
        Machado, que se relaizou no Colegio Nossa
37         Senhora de Lourdes, no Porto.
        </facto>
38 </foto>
39 <foto ficheiro="022-F-05.jpg">
40     <onde>Colegio Nossa Senhora de Lourdes, Porto</onde>
41     <quando data="1969-06-05"/>
42     <quem>Eugenia Maria de Oliveira Chamine Machado e uma
        Irma
43         do Colegio Nossa Senhora de Lourdes , no Porto
        .</quem>
44     <facto>Primeira comunhao de Eugenia Maria de Oliveira
        Chamine Machado.</facto>
45 </foto>
46 <foto ficheiro="022-F-06.jpg">
47     <quando data="1946-08-08"/>
48     <quem>Henrique Oliveira Machado, irmao de Antonio
        Machado</quem>

```

```

51      <facto> Comunhao solene de Henrique Oliveira Machado
          </facto>
52  </foto>
53  <foto ficheiro="022-F-10.jpg">
54      <quando>1977</quando>
55      <quem>Rosa Maria de Oliveira Chamine Machado</quem>
56      <facto>Comunhao solene de Rosa Maria de Oliveira
          Chamine Machado,
57          filha mais nova de Antonio Machado e Ana
          Chamine.</facto>
58  </foto>
59  <foto ficheiro="022-F-09.jpg">
60      <onde> Casa Machado, Afurada</onde>
61      <quando>1970</quando>
62      <quem>Ana de Lourdes Oliveira Chamine e Antonio
          Oliveira
63          Machado</quem>
64      <facto>Uma festa com alguns colegas de trabalho de
          Antonio
65          Machado.</facto>
66  </foto>
67  <foto ficheiro="022-F-08.jpg">
68      <onde>Casa Machado, Afurada</onde>
69      <quando data="2000-09-12"/>
70      <quem> Ana de Lourdes Oliveira Chamine</quem>
71      <facto>Dona Ana de Lourdes a fazer bolinhos de
          bacalhau para vender na
72          taberna "Casa Machado"</facto>
73  </foto>
74  <foto ficheiro="022-F-07.jpg">
75      <onde>Casa Machado, Afurada</onde>
76      <quem>Da esquerda para a direita: A prima Rosa
          Machado, a filha mais nova
77          Rosa Maria Machado, Antonio Machado, a filha
          Eugenia Machado, a tia
78          Isabel Coelho, o tio Manuel Moreira e a prima
          .</quem>
79      <facto>Reuniao de familia na Casa Machado</facto>
80  </foto>
81  <foto ficheiro="022-F-11.jpg">
82      <onde>Na praia de Nazare</onde>
83      <quando>1946</quando>
84      <quem> Antonio Oliveira Machado</quem>
85      <facto>Numa excursao a Fatima com o pai, fizeram uma
          paragem na
86          praia de Nazare.</facto>
87  </foto>
88  <foto ficheiro="022-F-12.jpg">
89      <quando>1940</quando>
90      <quem>Antonio Oliveira Machado</quem>

```

```

91      <facto>Comunhao solene de Antonio Machado, na Igreja
          de Santa
92          Marinha, Afurada.</facto>
93  </foto>
94  <foto ficheiro="022-F-02.jpg">
95      <onde>Casa Machado</onde>
96      <quando data="2000-09-12"/>
97      <quem>Ana de Lourdes Oliveira Chamine e Antonio
          Oliveira
98          Machado</quem>
99  </foto>
100 <foto ficheiro="022-F-14.jpg">
101     <quem>Ana de Lourdes Oliveira Chamine e Antonio
          Oliveira
102         Machado</quem>
103     <facto> Ana de Loudes tinha 18 anos e Antonio Machado
          22 anos. O
104         quadro foi uma prenda que Ana de Lourdes deu,
          pela altura do Natal, ao
105         marido</facto>
106 </foto>
107 <foto ficheiro="022-F-13.jpg">
108     <quem>Rosa Maria de Oliveira Chamine Machado e
          Eugenia Maria de
109         Oliveira Chamine</quem>
110     <facto> As duas filhas de Ant nio Machado. Rosa
          Maria tinha 5 e
111         Eugenia, 11 anos. </facto>
112 </foto>
113 <foto ficheiro="022-F-16.jpg">
114     <onde>Casa Machado</onde>
115     <quando data="2000-09-12"/>
116     <facto>O interior da taberna Casa Machado e alguns
          clientes, na sua maioria
117         pescadores.</facto>
118 </foto>
119 <foto ficheiro="022-F-17.gif">
120 <onde> Igreja Santa Marinha Afurada</onde>
121 <quando data="1961-01-15"/>
122 <quem> Ana de Lourdes Oliveira Chamine e Antonio Oliveira
          Machado</quem>
123 <facto>Casamento de Antonio Oliveira Machado com Ana de
          Lourdes Oliveira Chamine</facto>
124 <legenda> Antonio Oliveira Machado e Ana de Lourdes Oliveira
          Chamine, em frente
125 ao altar de Nossa Senhora de Fatima, na Igreja de Santa
          Marinha Afurada, no dia do casamento.</legenda>
126 </foto>
127 <foto ficheiro="022-F-15.jpg">
128 <onde>Casa Machado, Rua 27 de Fevereiro, Afurada </onde>

```

```

129 <quando data="2000-09-12"/>
130 <quem>Ana de Lourdes Oliveira Chamine e Antonio Oliveira
      Machado</quem>
131 </foto>
132
133
134 </fotos>

```

fg.xml

```

1 <?xml version="1.0"?>
2 <!DOCTYPE fotos SYSTEM "/Users/sony/mp/fotos.dtd">
3 <fotos>
4   <foto ficheiro="069-D-01.jpg">
5     <quando data="1969-05-12"> no quartel militar do
        Porto</quando>
6     <quem> Fausto Ferreira Gomes </quem>
7     <facto> Uma certid#227;o militar , foi o documento
        que foi exigido ao
8       senhor Fausto Ferreira Gomes para poder tirar
        a carta de condu#231;#227;o</facto>
9   </foto>
10  <foto ficheiro="069-F-01.jpg">
11    <onde>Taberna do Fausto , na Afurada</onde>
12    <quando data="2000-09-12"></quando>
13    <quem>Fausto Ferreira Gomes</quem>
14  </foto>
15  <foto ficheiro="069-F-02.jpg">
16    <onde>Taberna do Fausto , na Afurada</onde>
17    <quando data="2000-09-12"></quando>
18    <quem>Fausto Ferreira Gomes e um cliente</quem>
19    <facto>O interior da Taberna do Fausto situada na rua
        Alves Correia
20      n#186;145, Afurada.</facto>
21  </foto>
22  <foto ficheiro="069-F-03.jpg">
23    <onde>Monte de Santa Luzia , Viana do Castelo</onde>
24    <quando>1956</quando>
25    <quem>Fausto Ferreira Gomes e a esposa , Inf#226;ncia
        Silva
26      Magalh#227;es</quem>
27    <facto>Passeio do casal ao Monte de Santa Luzia em
        Viana do
28      Castelo.</facto>
29  </foto>
30  <foto ficheiro="069-F-04.jpg">
31    <onde>Taberna do Fausto , Afurada</onde>

```

```

32         <facto>Tabuleta, "Os alegres amigos do Fausto", foi
           oferecida ao senhor
33             Fausto pelos seus amigos e clientes.</facto>
34     </foto>
35     <foto ficheiro="069-F-05.jpg">
36         <onde>Taberna do Fausto, Afurada</onde>
37         <quem> Fausto Ferreira Gomes</quem>
38         <quando data="2000.09.22"></quando>
39         <legenda> Fausto Ferreira Gomes a preparar um "pirolito",
           uma mistura de vinho branco com 7up.</legenda>
40     </foto>
41     <foto ficheiro="069-F-06.jpg">
42         <quem>Fausto Ferreira Gomes</quem>
43         <quando data="2000.09.22"></quando>
44         <onde> Taberna Fausto, Afurada</onde>
45     </foto>
46 </fotos>

```

nc.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE fotos SYSTEM "../Users/sony/mp/fotos.dtd">
3 <fotos>
4     <foto ficheiro="026-F-01.jpg">
5         <onde>Taberna Neca Chamine, Afurada</onde>
6         <quando data="2000-09-12"/>
7         <quem>Manuel de Oliveira Chamine</quem>
8         <facto>Manuel de Oliveira Chamine atras do balcao na
           taberna
9             Neca Chamine.</facto>
10    </foto>
11    <foto ficheiro="026-F-02.jpg">
12        <onde>Taberna Neca Chamine, Afurada</onde>
13        <quando data="2000-09-12"/>
14        <quem>Manuel de Oliveira Chamine</quem>
15        <facto>O interior da taberna Neca Chamine situada na
           rua 27 de Fevereiro n197, Afurada.</facto>
16    </foto>
17 </fotos>

```

tf.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <!DOCTYPE fotos SYSTEM "Users/sony/mp/fotos.dtd">
3 <fotos>
4     <foto ficheiro="028-F-01.jpg">
5         <onde>Taberna do Fausto</onde>

```

```

6         <quando data="2000-09-22"/>
7         <quem>Jose Antonio Nunes da Silva , mais conhecido por
           "Matroco"</quem>
8     </foto>
9     <foto ficheiro="030-F-01.jpg">
10    <onde>Taberna do Fausto</onde>
11    <quando data="2000.09.22"/>
12    <quem>Julio Rodrigues Barbado, o pescador mais velho da
        Afurada.</quem>
13    </foto>
14    <foto ficheiro="030-F-02.jpg">
15    <onde>Taberna do Fausto</onde>
16    <quando data="2000.09.22"/>
17    <quem>J lio Rodrigues Barbado com um amigo.</quem>
18    </foto>
19    <foto ficheiro="031-F-01.jpg">
20    <onde>Taberna do Fausto</onde>
21    <quando data="2000.09.22"/>
22    <quem>Francisco de Sousa , cujo alcunha &acute; "Chico
        Frances".</quem>
23    </foto>
24    <foto ficheiro="031-F-02.jpg">
25    <onde>Taberna do Fausto</onde>
26    <quando data="2000.09.22"/>
27    <quem>Francisco de Sousa a beber um "pescador", que &acute;
        uma mistura de vinho tinto com 7up.</quem>
28    </foto>
29    <foto ficheiro="031-F-03.jpg">
30    <onde>Taberna do Fausto</onde>
31    <quando data="2000.09.22"/>
32    <quem>Francisco de Sousa com os amigos.</quem>
33    </foto>
34 </fotos>

```

A.0.9 Ficheiros Output

home.html

```

1 <!DOCTYPE html>
2 <head>
3     <meta char-set="utf8"/>
4 <title>Museu da Pessoa</title>
5     <style type="text/css">
6         #wrapper{
7             text-align:center;
8             margin-top:0px;
9             margin-bottom:0px;
10            padding:0px;

```



```

11     }          </style>
12 </head>
13 <body>
14     <div id="wrapper">
15         <nav>
16             <center><font size="7">&Iacutendice </
                font></center>
17             <ul>
18                 <p align="center" class="
                    scroll"><a href ="file:///
                    home/danielcaldas/Desktop/
                    Exerciciol/site/antonio.
                    html">Ant&oacutenio
                    Machado</a></p>
19                 <p align="center" class="
                    scroll"><a href ="file:///
                    home/danielcaldas/Desktop/
                    Exerciciol/site/fausto.
                    html">Fausto Gomes</a></p>
20                 <p align="center" class="
                    scroll"><a href ="file:///
                    home/danielcaldas/Desktop/
                    Exerciciol/site/taberna.
                    html">Taberna do Fausto</a
                    ></p>
21                 <p align="center" class="
                    scroll"><a href ="file:///
                    home/danielcaldas/Desktop/
                    Exerciciol/site/neca.html
                    ">Neca Chamin&eacute</a></
                    p>
22             </ul>          </nav>
23         <br><br>
24         <footer>
25             <p><b>Grupo de trabalho</b></p>
26             <p><i>Daniel Caldas a67691 / Marcelo
                Gon&ccedilalves a67736 / Ricardo
                Silva a67728</i></p>
27         </footer>
28     </div>
29 </body>
30 </html>

```

fausto.html

```

1 <!DOCTYPE html>
2 <head>
3     <meta char-set="utf8"/>

```

```

4 <title>Museu da Pessoa</title>
5   <style type="text/css">
6     #wrapper{
7       text-align:center;
8       margin-top:0px;
9       margin-bottom:0px;
10      padding:0px;
11    }
12  </head>
13  <body>
14    <div id="wrapper">
15      <nav>
16        <center><font size="10">Fausto Gomes</
17          font></center>
18        <center><font size="6">&Iacutendice </
19          font></center>
20        <ul>
21          <p align="center" class="
22            scroll"><a href="file:///
23              home/danielcaldas/Desktop/
24                Exerciciol/FaustoGomes/0
25                x8983210.html">Fausto
26                Ferreira Gomes</a></p>
27          <p align="center" class="
28            scroll"><a href="file:///
29              home/danielcaldas/Desktop/
30                Exerciciol/FaustoGomes/0
31                x8983830.html">Fausto
32                Ferreira Gomes e a esposa ,
33                Inf&#226;ncia Silva
34                Magalh&#227;es</a></p>
35          <p align="center" class="
36            scroll"><a href="file:///
37              home/danielcaldas/Desktop/
38                Exerciciol/FaustoGomes/0
39                x8983550.html">Fausto
40                Ferreira Gomes e um
41                cliente </a></p>
42        </ul>
43      </nav>
44      <br><br>
45    </div>
46  </body>
47 </html>

```

0x8983210.html

```

1 <!DOCTYPE html>
2 <head>

```

```

3      <meta char-set="utf8"/>
4 <title>Museu da Pessoa – Fausto Ferreira Gomes</title>
5      <style type="text/css">
6          #wrapper{
7              text-align:center;
8              margin-top:0px;
9              margin-bottom:0px;
10             padding:0px;
11         }      </style>
12 </head>
13 <body>
14     <div id="wrapper">
15         <p align="center"><font size=6>Fausto
16             Ferreira Gomes</font></p>
17         <br><br>
18         <hr color="black" size="3" width
19             ="80%">
20         <div id="069-D-01.jpg">
21             <ul>
22                 <p><b>Uma certidão
23                     militar, foi o documento
24                     que foi exigido ao senhor
25                     Fausto Ferreira Gomes para
26                     poder tirar a carta de
27                     condução</b></p>
28                 
31                 <p><i>12/5/1969</i></p>
32             </ul>
33             <hr color="black" size="3" width
34                 ="80%">
35         </div>
36         <div id="069-F-01.jpg">
37             <ul>
38                 <p><b>(sem título)</b></p>
39                 
42                 <p><i>12/9/2000</i></p>
43                 <p><b>Local</b>: Taberna do
44                     Fausto, na Afurada</i></p>
45             </ul>
46             <hr color="black" size="3" width
47                 ="80%">
48         </div>
49         <div id="069-F-05.jpg">
50             <ul>

```

```

36         <p><b>(sem t&iacutetulo)</b></
37         p>
         
         <p><i>22/9/2000</i></p>
39         <p>Fausto Ferreira Gomes a
           preparar um "pirolito",
           uma mistura de vinho
           branco com 7up.</p>
40         <p><i><b>Local</b>: Taberna do
           Fausto, Afurada</i></p>
41     </ul>
42     <hr color="black" size="3" width
         ="80%">
43 </div>
44 <div id="069-F-06.jpg">
45     <ul>
46         <p><b>(sem t&iacutetulo)</b></
47         p>
         
         <p><i>22/9/2000</i></p>
49         <p><i><b>Local</b>: Taberna
           Fausto, Afurada</i></p>
50     </ul>
51     <hr color="black" size="3" width
         ="80%">
52 </div>
53 </div>
54 </body>
55 </html>

```

Apêndice B

Código do Programa 2.5

B.0.10 filtrocancoes.l

```
1  espaco      [ \t ]*
2  novaMus     (\\#\\#)(\\-)+
3  titulo      {espaco}{?i:Title){espaco}:+
4  from        {espaco}{?i:From){espaco}:+
5  autor       {espaco}{?i:Author){espaco}:+
6  musica      {espaco}{?i:Music){espaco}:+
7  cantor      {espaco}{?i:Singer){espaco}:+
8  letra       {espaco}{?i:Lyrics){espaco}:+
9
10 %{
11     #include "List.h"
12     #include <stdlib.h>
13     #include <string.h>
14
15     List stack = NULL;
16     int i=0;
17     int z;
18
19     char *title = NULL, *author = NULL, *from = NULL;
20     char *lyrics = NULL, *singer = NULL, *music = NULL;
21     char **song = NULL;
22
23     char* trim(char* token);
24
25 %}
26
27 /* ----- ESTADOS
28      ----- */
29 %x  HEADER
30 %x  TEXT
```

```

31 %x   ESP
32
33 /*
    */
34
35
36 %%
37 BEGIN INITIAL;
38
39 { titulo }      {
40                  // A cada início de canção necessário
41                  alocar memória inicial para nova letra
42                  song = (char**) malloc(10*sizeof(char*));
43                  BEGIN HEADER;
44                  title = strdup(strchr(yytext, ':') + 1);
45                  title = trim(title);
46                  }
47 <HEADER>{autor}   if (!author) {
48                  author = strdup(strchr(yytext, ':') +
49                  1);
50                  author = trim(author);
51                  }
52 <HEADER>{musica}   if (!lyrics) {
53                  lyrics = strdup(strchr(yytext, ':') +
54                  1);
55                  lyrics = trim(lyrics);
56                  }
57 <HEADER>{cantor}   if (!singer) {
58                  singer = strdup(strchr(yytext, ':') +
59                  1);
60                  singer = trim(singer);
61                  }
62 <HEADER>{letra}     if (!music) {
63                  music = strdup(strchr(yytext, ':') + 1)
64                  ;
65                  music = trim(music);
66                  }
67 <HEADER>{from}      { BEGIN TEXT; }
68
69 <ESP>{novaMus}     {
70                  // Estado termina — filtrado ##——,
61                  inserimos nova canção na estrutura de
62                  dados e reinitializamos variáveis e
63                  condição de contexto.

```

```

71
72         stack = insertElemTail(stack, title, from,
73                                 author, lyrics, singer, music, song,
74                                 i);
75
76         for (z=0; z<i; z++){
77             free(song[z]);
78         }
79         free(song);
80
81         i=0;
82         BEGIN(INITIAL);
83
84         title = author = from = lyrics = singer =
85             music = NULL;
86
87     }
88
89     <TEXT>.+    {
90
91         song = (char **)realloc(song, (i+1)*sizeof(
92             char *));
93         song[i] = trim(strdup(ytext));
94         i++;
95         BEGIN ESP;
96     }
97
98     <ESP>.+    {
99
100         song = (char **)realloc(song, (i+1)*sizeof(
101             char *));
102         song[i] = trim(strdup(ytext));
103         i++;
104     }
105
106     <ESP>\n    {
107
108         if (i!=0){
109             song = (char **)realloc(song, (i+1)*
110                 sizeof(char *));
111             song[i] = trim(strdup(ytext));
112             i++;
113         }
114     }
115
116     <*>.\|\\n    { ; }
117
118 %%
119
120 /**
121  * Gera Latex a partir da estrutura de dados que resultou de
122  * filtrar o ficheiro
123  */

```

```

113 void latex(char *title, char *author, char *lyrics, char *
    singer, char *music, char **song, int size) {
114
115     char *aux;
116     aux = strdup(title);
117
118     FILE* f = fopen(strcat(aux, ".tex"), "w");
119
120     fprintf(f, "\\documentclass[12pt,twoside,a4paper]{article}
        \\n\\n");
121     fprintf(f, "\\usepackage[portuguese]{babel}\\n");
122     fprintf(f, "\\usepackage[utf8]{inputenc}\\n");
123     fprintf(f, "\\usepackage[T1]{fontenc}\\n");
124     fprintf(f, "\\usepackage[margin=3cm,nohead]{geometry}\\n");
125     fprintf(f, "\\newcommand{\\tab}[1]{\\hspace{.1\\textwidth}
        \\rlap{#1}}\\n");
126     fprintf(f, "\\renewcommand{\\baselinestretch}{1.2}\\n\\n");
127
128     fprintf(f, "\\begin{document}\\n\\n");
129
130     fprintf(f, "\\title{%s}\\n", title);
131
132     if(author){
133         fprintf(f, "\\author{%s}\\n\\n", author);
134     }
135     else{
136         if(!lyrics && music){
137             fprintf(f, "\\author{%s}\\n\\n", music);
138         }
139         else if(lyrics && !music){
140             fprintf(f, "\\author{%s}\\n\\n", lyrics);
141         }
142         else { fprintf(f, "\\author{%s \\and %s}\\n\\n", music,
            lyrics); }
143     }
144
145     fprintf(f, "\\date{\\n\\n}");
146
147     fprintf(f, "\\maketitle\\n\\n");
148
149     fprintf(f, "\\section*{Letra}\\n\\n");
150     for(z = 0; z < size; z++){
151         if(strcmp(song[z], "\\\\") != 0) { fprintf(f, "%s\\n",
            song[z]); }
152         else { fprintf(f, "%s", song[z]); }
153         //free(song[z]);
        //-----> Este
        free est a causar double free corruption...
154     }
155

```



```

156     if(singer){
157         fprintf(f, "\\begin{flushright}\\n");
158         fprintf(f, "%s\\n", singer);
159         fprintf(f, "\\end{flushright}\\n");
160     }
161
162     fprintf(f, "\\n \\end{document}\\n");
163     if(f) fclose(f);
164 }
165
166 /**
167  * Gera Latex a partir da estrutura de dados que resultou de
168   * filtrar o ficheiro
169  */
170 void gerarLatex()
171 {
172     int size = 0;
173     List aux;
174     aux=stack;
175
176     while(aux){
177         title = getTitle(aux); author = getAuthor(aux); lyrics
178             = getLyrics(aux);
179         singer = getSinger(aux); music = getMusic(aux); song =
180             getSong(aux);
181         size = getSize(aux);
182
183         latex(title, author, lyrics, singer, music, song, size
184             );
185         title=author=lyrics=singer=music=NULL;
186         song=NULL;
187         size=0;
188         aux = nextElem(aux);
189     }
190 }
191
192 char* trim(char* token) {
193     int i=0, n;
194     if(token[i]=='\\n'){
195         n=strlen(token)+2;
196         token = (char*)realloc(token, n*sizeof(char));
197         token[i]='\\';
198         token[i+1]='\\';
199         token[i+2]='0';
200     } else {
201         for(i=0; token[i]!=' ' || token[i]!='\\t'; i++) ;
202         token+=i;
203
204         n = strlen(token);
205     }

```

```

202         for(i=n-1; token[i]==' ' || token[i]=='\t'; i--) ;
203         token[i+1]='\0';
204     }
205     return token;
206 }
207
208 int yywrap() {
209     gerarLatex();
210     freeList(stack);
211     return 1;
212 }

```

B.0.11 List.c

```

1 #include "List.h"
2
3 /*      Estrutura de dados      */
4
5 typedef struct Element {
6
7     char *title;
8     char *from;
9     char *author;
10    char *lyrics;
11    char *singer;
12    char *music;
13    char **song;
14    int size;
15
16    struct Element *next;
17 } LinkedList;
18
19 /* Função que cria um elemento novo para a lista */
20 List newElem() {
21     LinkedList *new = (LinkedList *) malloc(sizeof(LinkedList))
22         ;
23
24     new->title = NULL;
25     new->from = NULL;
26     new->author = NULL;
27     new->lyrics = NULL;
28     new->singer = NULL;
29     new->music = NULL;
30     new->song = (char **) malloc(sizeof(char *));
31     new->next = NULL;
32     new->size = 0;
33
34     return new;

```

```

34 }
35
36 /* Funções que retiram de um elemento a informação */
37 void *getTitle (List elem) {
38     if (!elem) return NULL;
39     return elem->title;
40 }
41
42 /* Retorna a origem da canção */
43 void *getFrom (List elem) {
44     if (!elem) return NULL;
45     return elem->from;
46 }
47
48 /* Retorna o Autor da canção */
49 void *getAuthor (List elem) {
50     if (!elem) return NULL;
51     return elem->author;
52 }
53
54 /* Retorna o autor da letra da canção */
55 void *getLyrics (List elem) {
56     if (!elem) return NULL;
57     return elem->lyrics;
58 }
59
60 /* Retorna o cantor da canção */
61 void *getSinger (List elem) {
62     if (!elem) return NULL;
63     return elem->singer;
64 }
65
66 /* Retorna o Autor da música */
67 void *getMusic (List elem) {
68     if (!elem) return NULL;
69     return elem->music;
70 }
71
72 /* Retorna a Letra da música */
73 void* getSong (List elem) {
74     if (!elem) return NULL;
75     return elem->song;
76 }
77
78 /* Retorna o tamanho da letra */
79 int getSize (List list){
80     if (!list) return 0;
81     return list->size;
82 }
83

```

```

84 /* Verifica se existe pr ximo elemento */
85 int hasNext (List list) {
86     if (!list) return 0;
87     return (list->next != NULL);
88 }
89
90 /* Fun o que insere um novo elemento na cauda da lista */
91 List insertElemTail(List list, char *title, char *from, char *
    author, char *lyrics, char *singer, char *music, char **
    song, int size) {
92     LinkedList *elem = newElem();
93     LinkedList *aux;
94     int x;
95
96     if(title != NULL) elem->title = strdup(title);
97     if(from != NULL) elem->from = strdup(from);
98     if(author != NULL) elem->author = strdup(author);
99     if(lyrics != NULL) elem->lyrics = strdup(lyrics);
100    if(singer != NULL) elem->singer = strdup(singer);
101    if(music != NULL) elem->music = strdup(music);
102
103    elem->song = (char **)realloc(elem->song, size*sizeof(char
        *));
104    for(x = 0; x < size; x++){
105        elem->song[x] = strdup(song[x]);
106    }
107
108    elem->size = size;
109
110    if (!list) return elem;
111    else {
112        aux = list;
113        while (aux->next) aux = aux->next;
114        aux->next = elem;
115    }
116    return list;
117 }
118
119 /* Fun o que insere um novo elemento na cabe a da lista */
120 List insertElemHead(List list, char *title, char *from, char *
    author, char *lyrics, char *singer, char *music, char **
    song, int size) {
121    LinkedList *elem = newElem();
122    int x;
123
124    if(title != NULL) elem->title = strdup(title);
125    if(from != NULL) elem->from = strdup(from);
126    if(author != NULL) elem->author = strdup(author);
127    if(lyrics != NULL) elem->lyrics = strdup(lyrics);
128    if(singer != NULL) elem->singer = strdup(singer);

```

```

129     if (music != NULL)     elem->music = strdup (music);
130
131     elem->song = (char **) realloc (elem->song, size*sizeof (char
132                                     *));
133     for (x = 0; x < size; x++){
134         elem->song[x] = strdup (song[x]);
135     }
136     elem->size = size;
137
138     elem->next = list;
139     return elem;
140 }
141
142 /* Retorna o proximo elemento da lista */
143 List nextElem (List elem) {
144     if (!elem) return NULL;
145     return elem->next;
146 }
147
148 /* Retorna o tamanho da lista */
149 int listSize (List list) {
150     int n = 0;
151     LinkedList *aux = list;
152     while (aux) {
153         aux = aux->next;
154         n++;
155     }
156     return n;
157 }
158
159 /* Liberta a memoria ocupada pela lista */
160 void freeList (List list) {
161     LinkedList *aux;
162
163     while (list!=NULL) {
164         aux = list;
165         list = list->next;
166         free (aux);
167     }
168 }

```

B.0.12 List.h

```

1 #ifndef _List
2 #define _List
3
4 #include <stdio.h>

```

```

5 #include <stdlib.h>
6 #include <string.h>
7
8 /** Estruturas de dados */
9 typedef struct Element *List;
10
11 /* Função que cria um elemento novo para a lista */
12 List newElem();
13
14 /* Funções que retiram de um elemento a informação */
15 void *getTitle (List elem);
16
17 /* Retorna a origem da canção */
18 void *getFrom (List elem);
19
20 /* Retorna o Autor da canção */
21 void *getAuthor (List elem);
22
23 /* Retorna o autor da letra da canção */
24 void *getLyrics (List elem);
25
26 /* Retorna o cantor da canção */
27 void *getSinger (List elem);
28
29 /* Retorna o Autor da música */
30 void *getMusic (List elem);
31
32 /* Retorna a Letra da música */
33 void* getSong (List elem);
34
35 /* Retorna o tamanho da letra */
36 int getSize (List list);
37
38 /* Verifica se existe próximo elemento */
39 int hasNext (List list);
40
41 /* Função que insere um novo elemento na cauda da lista */
42 List insertElemTail(List list, char *title, char *from, char *
    author, char *lyrics, char *singer, char *music, char **
    song, int size);
43
44 /* Função que insere um novo elemento na cabeça da lista */
45 List insertElemHead(List list, char *title, char *from, char *
    author, char *lyrics, char *singer, char *music, char **
    song, int size);
46
47 /* Retorna o próximo elemento da lista */
48 List nextElem (List elem);
49
50 /* Retorna o tamanho da lista */

```

```
51 int listSize (List list);
52
53 /* Liberta a memoria ocupada pela lista */
54 void freeList (List list);
55
56 #endif
```

B.0.13 runcancoes.sh

```
1 #!/bin/bash
2 mkdir resultados
3 cat teste/Teste.lyr | ./Latex.exe
4 for file in *.tex; do pdflatex "
```

B.0.14 clearcancoes.sh

```
1 #!/bin/bash
2 rm *.tex
3 rm *.log
4 rm *.aux
```

Apêndice C

Código do Programa 2.2

C.0.15 enamex.l

```
1 pessoa \<ENAMEX\ TYPE\="PERSON"\>.+</ENAMEX\>
2 cidade \<ENAMEX\ TYPE\="LOCATION"[ \t]*(SUBTYPE\="CITY")
   ?\>.+</ENAMEX\>
3 pais \<ENAMEX\ TYPE\="LOCATION"[ \t]*(SUBTYPE\="COUNTRY")
   ?\>.+</ENAMEX\>
4 org \<ENAMEX\ TYPE\="ORGANIZATION"\>.+</ENAMEX\>
5
6
7 %{
8     #include <stdio.h>
9     #include <string.h>
10
11     char *nome;
12     char* cidade;
13     char* pais;
14     char* org;
15
16     /**
17      * Função que retira espaços do início e fim de uma
18      string.
19      *@param recebe uma string.
20      *@return retorna apontador para string modificada.
21      */
22     char* trim(char* token)
23     {
24         int i, n;
25
26         for(i=0; token[i]!=' ' || token[i]!='\t'; i++)
27             ;
28         token+=i;
29         n = strlen(token);
```



```

29         for ( i=n-1; token[i]!=' ' || token[i]!='\t'; i
30             --)
31             ;
32         token[i+1]='\0';
33         return token;
34     }
35
36     char* extract_value (char* token)
37     {
38         int i, n;
39
40         token = strchr(token, '>');
41         token++;
42         for ( i=strlen(token)-1; token[i]!='<' && i>=0;
43             i--)
44             ;
45         token[i]='\0';
46         token = trim(token);
47     }
48     %}
49
50
51
52
53     %%
54
55     {pessoa} {
56         nome = strdup(extract_value(yytext));
57         printf("NOME: [%s]\n", nome);
58     }
59
60     {cidade} {
61         cidade = strdup(extract_value(yytext));
62         printf("CIDADE: [%s]\n", cidade);
63     }
64
65     {pais} {
66         pais = strdup(extract_value(yytext));
67         printf("PAIS: [%s]\n", pais);
68     }
69
70     {org} {
71         org = strdup(extract_value(yytext));
72         printf("ORG: [%s]\n", org);
73     }
74 }
75
76 <*>.\n { ; }

```

APÊNDICE C. CÓDIGO DO PROGRAMA 2.2

⁷⁷
⁷⁸ %%
