Processamento de Linguagens

Trabalho Prático 1

Relatório de Desenvolvimento

Grupo 31

António Lindo	Nuno Cunha	Pedro Parente
(A85813)	(A85400)	(A85919)

5 de Abril de 2020

Resumo

Este relatório é referente ao trabalho prático 1 realizado no ambito da cadeira de processamento de linguagem. Foram fornecidos aos alunos 5 enunciados dos quais o nosso grupo escolheu resolver o primeiro que consiste no processamento de um template multi-file.

Conteúdo

Capítulo 1

Introdução

1.1 Enquadramento e Problema

É normal, na maior parte dos projetos de software haverem soluções que envolvam vários ficheiros e diretorias. Por exemplo um projeto pode conter um ficheiro com o código fonte, uma Makefile, uma diretoria com exemplos etc. Todos estes ficheiros estarão dentro de uma diretoria principal do projeto. O problema deste projeto passa por arranjar um método de através de um template multi-file facilitar todo este processo de criação de ficheiros e diretorias.

1.2 Objetivos

O objetivo deste trabalho prático seria a implementação de um programa "mkfromtemplate". Este programa receberia um nome de um projeto e um ficheiro template multi-file com as diretorias e ficheiros a criar e ainda o respetivo conteúdo de cada ficheiro. Através destes dois argumentos, o programa seria capaz de criar todos os ficheiros e diretorias e ainda preencher cada ficheiro com o seu conteúdo.

1.3 Estrutura do relatório

No capítulo 2 do relatório é feita uma breve descrição do problema e ainda uma análise aos requisitos que este projeto nos impôs. No capítulo 3 são explicadas todas as decisões que tomamos na implementação do programa em questão e especificadas estruturas de dados e algoritmos que utilizamos. No capítulo 4 são demonstrados vários exemplos de testes para o nosso programa e os respetivos resultados e no capítulo 5 é feita uma conclusão ao trabalho.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

O problema proposto pelo Professor baseia-se na simplificação de criação de projetos com múltiplos ficheiros e diretorias com uma hierarquia específica.

2.2 Especificação do Requisitos

Para a resolução do problema foram-nos propostos os seguintes requisitos para implementar na nossa solução:

- Definir Expressões Regulares que encontrem os padrões usados para a criação do projeto;
- Construir a hierarquia de pastas definida na secção tree;
- Criar os ficheiros definidos na secção tree nas diretorias corretas;
- Tanto na criação de ficheiros como no conteúdo destes, substituir o name, author e email pelos respetivos valores guardados na fase de meta-dados e execução do comando que cria o projeto;

Capítulo 3

Concepção/desenho da Resolução

3.1 Estruturas de Dados

Neste trabalho a unica estrutura de dados de que fizemos uso foi uma hash table para guardar a localização de Ficheiros. Os restantes algoritmos foram realizados apenas com recurso a variáveis e arrays.

3.2 Algoritmos

Os algoritmos usados que queremos destacar são os de criação da diretoria e de escrita nos ficheiros. Relativamente à criação de diretorias, fizemos uso de uma variavel char * currentDirectory. Esta variavel como o nome indica mantem (à medida que é precorrida a tree incluida no template) o path do diretório em que devem ser criadas as diretorias/ficheiros subsequentes.

```
1  void removeDirectory(char *t, int tracos){
2    int pos = strlen(t) - 2;
3    while(tracos > 0){
4         if(t[pos] == '/') {tracos--;}
5         else {t[pos] = '\0';}
6         pos--;
7    }
8  }
9
10 // char * d == yytext (cada linha da tree)
11 void updateDirectory(char *d){
12   int tracos = contaTracos(d);
13   if(tracos > currentTracos){
14     strcat(currentDirectory, d + tracos + 1);
15   }
16   else{
17     removeDirectory(currentDirectory, currentTracos - tracos + 1);
18     strcat(currentDirectory, d + tracos + 1);
19   }
20   currentTracos = tracos;
21 }
```

Isto é feito tendo em conta a variação do numero hífens de uma linha da tree para a seguinte.

Relativamente à escrita dos ficheiros, quando entramos numa start condition de escrita é lido caracter a caracter e é escrito tudo no ficheiro cuja localização foi guardada numa hastable no momento que foi percorrida a tree.

```
1  void writeToFile(const char* f){
2    FILE * file = fopen(currentFile, "a+");
3    printf("Printing to file: %s\n", f);
4    fputs(f, file);
5    fclose(file);
6  }
7
8  <TREE>^-[-]*[ ]\{%name%\}\..*          {int trac = contaTracos(yytext);
9
10                                        name = malloc(sizeof(char) * 30);
11
12                                        strcpy(name, nome);
13
14                                        char *extension = yytext + trac + 1 + 8;
15
16                                        strcat(name, extension);
17
18                                        updateDirectoryCoragem(yytext, name);
19
20                                        touch(name);
21
22                                        mv(name, currentDirectory);
23
24                                        char* fileDirectory = strdup(currentDirectory);
25
26                                        //funcao que insere a diretoria na hashtable
27                                        g_hash_table_insert(files,
28                                        (gpointer) yytext + currentTracos + 1,
29                                        (gpointer) fileDirectory);
30                                        }
```

Capítulo 4

Codificação e Testes

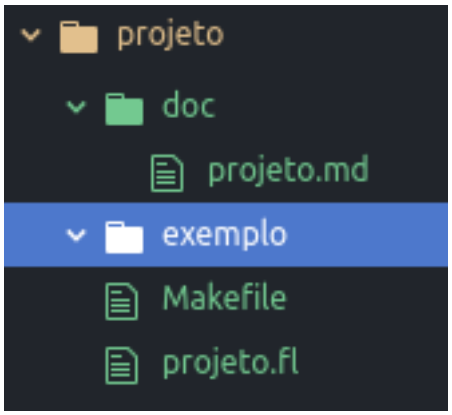
4.1 Testes realizados e Resultados

No primeiro teste utilizamos como nome do projeto: projeto e o seguinte template multi-file:

```
1 === meta
2
3 email: jj@di.uminho.pt
4 author: J.Joao
5 # "name"     dado por argumento de linha de comando (argv[1])
6
7 === tree
8 {%name%}/
9 - {%name%}.fl
10 - doc/
11 --{%name%}.md
12 - exemplo/
13 - Makefile
14
15 === Makefile
16
17 {%name%}: {%name%}.fl
18         flex {%name%}.fl
19         cc -o {%name%} lex.yy.
20
21 install: {%name%}
22         cp {%name%} /usr/local/bin/
23
24 === {%name%}.md
25 # NAME
26 {%name%} - o nosso fabuloso filtro   ...FIXME
27
28 ## Synopsis
29
30   {%name%} file*
31
32 ## Description
33 ## See also
34 ## Author
35
```

```
36 Comments and bug reports to {%author%}, {%email%}.
37
38 ==== {%name%}.fl
39 %option noyywrap yylineno
40 %%
41
42 %%
43 int main(){
44   yylex();
45
46   return 0;
47   }
```

Executando o comando ./mkfromtemplate projeto ¡ template.txt, obtivemos os seguintes resultados:
Diretorias e ficheiros criados:



Conteúdo do ficheiro projeto.md:

```
1 # NAME
2 projeto - o nosso fabuloso filtro  ...FIXME
3
4 ## Synopsis
5
6   projeto file*
7
8 ## Description
9 ## See also
10 ## Author
11
12 Comments and bug reports to J.Joao, jj@di.uminho.pt.
```

Conteúdo do ficheiro projeto.fl:

```
1 %option noyywrap yylineno
2 %%
3
4 %%
5 int main(){
6   yylex();
7
8   return 0;
9   }
```

Conteúdo do ficheiro Makefile:

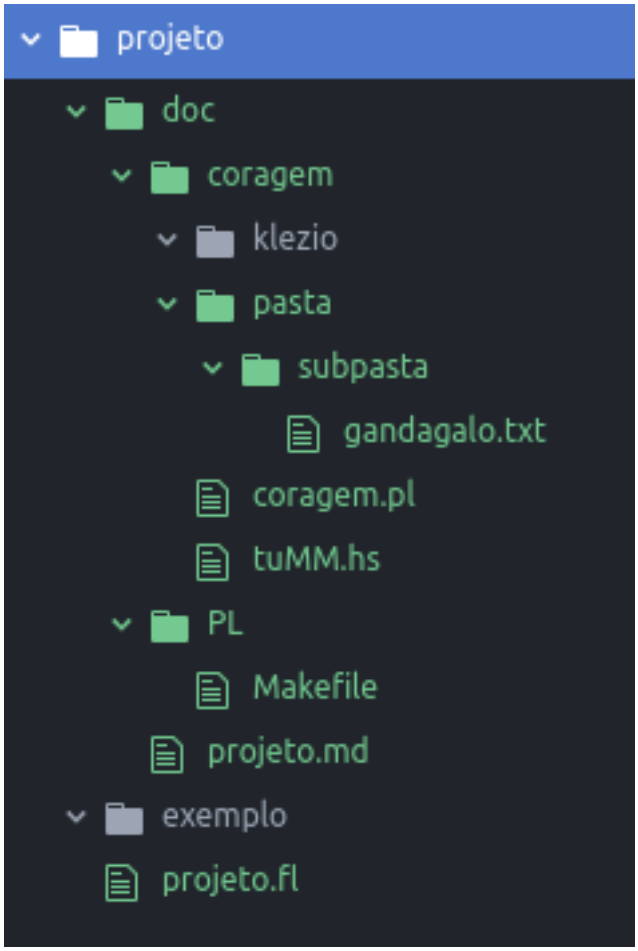
```
projeto: projeto.fl
	flex projeto.fl
	cc -o projeto lex.yy.

install: projeto
	cp projeto /usr/local/bin/
```

No segundo teste, utilizamos um template multi-file mais complexo:

```
=== meta

email: jj@di.uminho.pt
author: J.Joo
# "name"     dado por argumento de linha de comando (argv[1])

=== tree
{%name%}/
- {%name%}.fl
- doc/
-- coragem/
--- coragem.pl
--- pasta/
---- subpasta/
----- gandagalo.txt
--- tuMM.hs
--- klezio/
-- PL/
--- Makefile
-- {%name%}.md
- exemplo/

=== Makefile

{%name%}: {%name%}.fl
	flex {%name%}.fl
	cc -o {%name%} lex.yy.

install: {%name%}
	cp {%name%} /usr/local/bin/

=== coragem.pl

Durante a antiguidade, Chronos era ocasionalmente confundido com o tit  Cronos. De acordo com Plutarco, os gregos antigos acreditavam que Cronos era um nome aleg rico para Chronos. O que quer dizer que, na verdade, a figura de Chronos era , fundamentalmente, a mesma que a do tit  Cronos, o deus do tempo da teogonia hesi dica e do culto comum dos gregos.

Al m do nome, a hist ria de Cronos comer seus filhos tamb m era interpretada como uma alegoria de um aspecto espec fico do tempo, a esfera de influ ncia de Chronos. Chronos representava as caracter sticas destrutivas de tempo, que consumia todas as coisas, um conceito que foi definitivamente ilustrado quando o rei tit  consumiu os deuses do Olimpo     o passado consumindo o futuro, a
```

```
    gera   o mais velha suprimida pela gera   o seguinte.

=== tuMM.hs

myKlezio :: Int a -> [char]
myKlezio 0 = ['k','e','z','i','o']
myKlezio _ = ['d','j','k','l','e','z','i','o']

=== {%name%}.md
# NAME
{%name%} - o nosso fabuloso filtro   ...FIXME

## Synopsis

  {%name%} file*

## Description
## See also
## Author

Comments and bug reports to {%author%}, {%email%}.

=== {%name%}.fl
%option noyywrap yylineno
%%

%%
int main(){
  yylex();

  return 0;
  }
```

Executando o comando ./mkfromtemplate projeto ¡ template2.txt todos os ficheiros ficaram com os conteúdos definidos tal como estava no template e obtemos o seguinte resultado:



Capítulo 5

Conclusão

Terminado este trabalho prático da unidade curricular de processamento de linguagens, podemos concluir que a realização do mesmo contribuíu muito para a consolidação da matéria lecionada nas aulas, em particular a utilização de Expressões regulares (ERs) para fazer filtros de textos. Quanto ao trabalho em si, estamos satisfeitos com o resultado final, uma vez que, este é capaz de realizar todos os objetivos propostos no enunciado.

Apêndice A

Código do Programa

```
1  %{
2  #include <string.h>
3  #include <stdlib.h>
4  #include <glib.h>
5
6
7  const char* nome;
8  char* name;
9  char *email;
10 char *author;
11 char currentDirectory[200] = "";
12 int currentTracos = -1;
13 GHashTable* files;
14 char* currentFile;
15
16 void mv(const char* source, const char* dest){
17   char com[100];
18   sprintf(com, "mv %s %s", source, dest);
19   system(com);
20 }
21
22 void mkdir(const char* d){
23   char com[100];
24   sprintf(com, "mkdir %s", d);
25   system(com);
26 }
27
28 void touch(const char *d, ...){
29   char com[100];
30   sprintf(com, "touch %s", d);
31   system(com);
32 }
33
34 int contaTracos(char *t){
35   int ti = 0;
36
37   for(int i = 0; i < strlen(t); i++){
38     if('-' == t[i]) ti++;
39   }
```

```
40
41   return ti;
42 }
43
44 void removeDirectory(char *t, int tracos){
45   int pos = strlen(t) - 2;
46   while(tracos > 0){
47       if(t[pos] == '/') {tracos--;}
48       else {t[pos] = '\0';}
49       pos--;
50   }
51 }
52
53 void updateDirectory(char *d){
54   int tracos = contaTracos(d);
55   if(tracos > currentTracos){
56     strcat(currentDirectory, d + tracos + 1);
57   }
58   else{
59     removeDirectory(currentDirectory, currentTracos - tracos + 1);
60     strcat(currentDirectory, d + tracos + 1);
61   }
62   currentTracos = tracos;
63 }
64
65 void updateDirectoryCoragem(char *d, char* name){
66   int tracos = contaTracos(d);
67   if(tracos > currentTracos){
68     strcat(currentDirectory, name);
69   }
70   else{
71     removeDirectory(currentDirectory, currentTracos - tracos + 1);
72     strcat(currentDirectory, name);
73   }
74   currentTracos = tracos;
75 }
76
77 void writeToFile(const char* f){
78   FILE * file = fopen(currentFile, "a+");
79   fputs(f, file);
80   fclose(file);
81 }
82
83 gboolean hashCompare(gconstpointer s1, gconstpointer s2) {
84
85   if(g_strcmp0(s1,s2) != 0){
86     return TRUE;
87   }
88
89   return FALSE;
90
91 }
```

```
%}
%option noyywrap

%x META TREE DIRECTORY WRITE

%%

<*>^\=\=                                {BEGIN INITIAL; }

\=[ ]meta                               {BEGIN META; }

<META>^email:[ ].*                      {email = strdup(yytext + 7); }

<META>^author:[ ].*                     {author = strdup(yytext + 8); }

\=[ ]tree                               {BEGIN TREE; }

<TREE>^\{%name%\}\/                     {strcat(currentDirectory, nome);
                                         strcat(currentDirectory,"/");
                                         mkdir(currentDirectory);
                                        }

<TREE>^[-]*[ ][A-Za-z0-9]*\/            {updateDirectory(yytext);
                                         mkdir(currentDirectory);
                                        }

<TREE>^-[-]*[ ]\{%name%\}\..*           {int trac = contaTracos(yytext);
                                         name = malloc(sizeof(char) * 30);
                                         strcpy(name, nome);
                                         char *extension = yytext + trac + 1 +
                                             8;
                                         strcat(name, extension);
                                         updateDirectoryCoragem(yytext, name);
                                         touch(name);
                                         mv(name, currentDirectory);
                                         char* fileDirectory = strdup(
                                             currentDirectory);
                                         g_hash_table_insert(files, (gpointer)
                                             yytext + currentTracos + 1, (
                                             gpointer) fileDirectory);
                                        }

<TREE>^-[-]*[ ][A-Za-z0-9\.]*
```

```
\=[ ].*                                          {BEGIN WRITE;
                                                  char* fileName = strdup(yytext + 2);
                                                  currentFile = g_hash_table_lookup(
                                                      files, fileName);
                                                 }

<WRITE>.|\n                                      {writeToFile(yytext);}

<WRITE>\{%name%\}                                {writeToFile(nome);}

<WRITE>\{%author%\}                              {writeToFile(author);}

<WRITE>\{%email%\}                               {writeToFile(email);}



%%

int main(int argc, char const *argv[]){
  nome = argv[1];
   files = g_hash_table_new(g_str_hash , hashCompare);
   yylex();
   return(0);
}
```