Processamento de Linguagens e Compiladores (3º ano LCC)

Transformador Publico2NetLang

Relatório de Desenvolvimento

André Morandi
A86912

Hélder Lopes
A87951

Novembro 2020

Resumo

Na disciplina de Processamento de Linguagens e Compiladores, foi apresentado o analisador FLex, que possibilita escrever padrões e scanners com Expressões Regulares para desenvolver filtros em vários tipos de ficheiros. Em conjunto com o GCC, transformamos o código extraído em programas executáveis. Para este projeto, vamos fazer um filtro FLex que permita buscar dados relevantes na secção de comentários de uma dada notícia de Jornal, para depois serem analisados.

Conteúdo

Capítulo 1

Introdução

Supervisor: Prof. Pedro Rangel Henriques

1.1 Contextualização

Área: Processamento de Linguagens

Foi nos apresentado o analisador FLex, que possibilita escrever padrões e scanners com Expressões Regulares para desenvolver filtros em vários tipos de ficheiros. Em conjunto com o GCC, transformamos o código extraído em programas executáveis.

1.2 Objetivos

Com o objetivo de se fazer um estudo sócio-linguístico da forma e conteúdo dos comentários de uma notícia de Jornal (Jornal Público), pretende-se fazer um filtro FLex que permita buscar dados relevantes, como nome do utilizador, data e texto do comentário,e os converta de HTML para o formato JSON. Ao fim do projeto, nosso analisador deverá conseguir filtrar e gerar o ficheiro para qualquer página de comentários encontrada neste Jornal. O resultado final será armazando da seguinte forma:

```
"commentThread": [
    {
        "id": "STRING",
        "user": "STRING",
        "date": "STRING",
        "timestamp": NA,
        "commentText": "STRING",
        "likes": NUMBER,
        "hasReplies": TRUE/FALSE,
        "numberOfReplies": NUMBER
        "replies": [ ]
    },.......
]
```

Capítulo 2

Resolução

2.1 Filtros base

A parte inicial do projeto é declarar as Expressões Regulares (ER) que serão utilizadas para filtrar as partes úteis do código HTML (texto fonte). ERs que representam sequências de letras e números para filtrar ID do comentário, ID de utilizador e a data (dd-mm-yyyy).

```
%}

digit [0-9]

alphanum [0-9a-zA-Z]

id_formatt {alphanum}{8}-{alphanum}{4}-{alphanum}{4}-{alphanum}{4}-{alphanum}{12}

date_formatt {digit}{2}\.{digit}{2}\.{digit}{4}

%%
```

2.2 Filtros de tags HTML

A seguir, definimos padrões que representam as tags de HTML principais, onde se encontram informações necessárias para a análise pedida.

```
%}

com_section_init         li [’  ’]+class=\"comment\" [’  ’]+
com_section_fim         \<\ /li\>

com_list_init            \<ol [’  ’]+class=\"comments__list\"
com_list_fim            \<\ / ol\>
```

```
com_content_init      \<p\>['\n']*['  ']*(\t)*
com_content_fim      ['  ']*(\t)*\<\/p\>

user_info_init           \/utilizador\/perfil\/
user_info_fim            \<\/a\>

date_info_init           time['  ']+class=\"dateline['  ']+comment__dateline\"
date_info_fim            \/time\>

com_id_init              data-comment-id=
com_id_fim               \>

%%
```

2.3 Start Conditions

Pela nosso entendimento do trabalho, achamos mais adequado utilizar `start conditions` (estados) inclusivos, ou seja, as regras que não tem um estado definido, estão sempre ativas, enquanto as outras só estão ativas de acordo com a especificação da linha. Os estados inclusivos são representados por %s.
Estes foram os estados definidos pelo grupo:

```
%s COMENTARIO    USER    TEXTO    ID_COMENTARIO    DATA
```

2.4 Stacks

Para podermos alternar os estados ativos em um determinado momento, utilizamos o recurso de stacks (Last In First Out). Com isto, quando quisermos que um estado esteja ativo, adicionamos ao topo da stack: yy_push_state(ESTADO). Ao sair deste estado, tiramos do topo, analogamente: yy_pop_state( ). Para o FLex reconhecer estas funções, foi necessário adicionar %option stack à primeira secção do ficheiro
Exemplo e demonstração do processo na stack usado na nossa resolução:

```
<COMENTARIO>{user_info_init} { yy_push_state(USER); }
<USER>{id_formatt}       { (*frente)userid =strdup( yytext); }
<USER>{user_info_fim}    {  yy_pop_state( ); }
```

Quando encontramos, dentro de um comentário, a tag que determina aonde está o ID do utilizador, colocamos a start condition USER no topo da stack (yy_push_state(USER);). Como a start condition USER está agora ativa, procuramos pelo ID do utilizador, através do filtro {id_formatt} e guardamo-os. Quando encontramos a tag que determina o fim das informações do utilizador({user_info_fim}),tiramos da stack (yy_pop_state( );)

2.5 Código C e estrutura de dados

Para podemos andar pelo ficheiro e individualizar cada comentário, criamos uma struct que recorre ao conceito de lista ligada. As informações contidas na struct incluem: ID do comentário, ID do utilizador,

data do comentário, o texto (conteúdo) do comentário, um contador de profundidade para saber se é um comentário principal ou resposta (será explicado mais à frente neste relatório) e uma instância da struct chamada recursivamente para representar o próximo comentário da sequência.

```
typedef struct commentThread {
    char *id;
    char *userid;
    char *date;
    char *commentText;
    struct commentThread *prox;
    int count;
} *Comment;
```

A função a seguir é usada para quando queremos fazer um novo comentário (abrir espaço na memória). O retorno da função é do tipo apontador para um comentário (Comment).

```
Comment newComment (int x) {
    Comment l = malloc (sizeof (struct commentThread ) );
    lprox = NULL;
    lcount = x;

    return l;
}
```

Criamos uma variável e um apontador do tipo Comment, para nos auxiliar enquanto lemos o ficheiro.

```
Comment inicio, *frente;
```

Sendo que a variável inicio vai guardar a primeira posição da lista ligada enquanto o apontador *frente vai preenchendo a lista com as devidas informações retiradas do ficheiro html. Sempre que encontramos uma abertura da tag 'ol' (lista ordenada), o qual se refere a uma lista de comentários, somamos um ao contador (variável int numero). Quando chegamos ao final da tag 'ol', subtraímos um. Permitindo assim, guardar para cada comentário a sua respetiva profundidade em relação aos comentários anteriores. Se o contador estiver em 1, significa que o comentário em questão é principal. Se o contador for maior que 1, então este é uma resposta ao último comentário com o valor menor do que o atual, possibilitando a existência de respostas de respostas.

```
%%

{com_list_init}         {  numero++;  }
{com_list_fim}          {  numero--;  }

{com_section_init}   {  yy_push_state(COMENTARIO); frente=&((*frente)prox);
(*frente)=newcomment(numero);  }

<COMENTARIO>{com_id_init}  {  yy_push_state(ID_COMENTARIO);  }
```

```
<COMENTARIO>{user_info_init}  {  yy_push_state(USER);  }
<COMENTARIO>{date_info_init}    {  yy_push_state(DATA);  }
<COMENTARIO>{com_content_init}  {  yy_push_state(TEXTO); }
<COMENTARIO>{com_section_fim}   {  yy_pop_state();  }

<ID_COMENTARIO>{id_formatt} { (*frente)id =strdup( yytext); }
<ID_COMENTARIO>{com_id_fim} { yy_pop_state(); }

<USER>{id_formatt}  {  (*frente) userid =strdup( yytext); }
<USER>{user_info_fim}   {  yy_pop_state(); }

<DATA>{date_formatt} { (*frente)date =strdup( yytext); }
<DATA>{date_info_fim}  {  yy_pop_state(); }

<TEXTO>[^\<]*    {  (*frente)commentText =strdup( yytext); }
<TEXTO>{com_content_fim}        {  yy_pop_state(); }

(.|\n)                      {   ;  }

%%
```

Capítulo 3

Conclusão

3.1 Resumo Final

Resumidamente, nosso projeto consiste em ler um ficheiro HTML de comentários, através do FLex, armazenar o seu conteúdo numa lista ligada e criar um JSON com as informações relevantes de cada comentário.

3.2 Dificuldades

Ao longo do projeto tivemos algumas dúvidas mas nenhuma teve grande impacto, pois rapidamente achávamos solução à exceção dos apontadores em C, os quais eram necessários para entender a melhor maneira de usar as listas ligadas e a administração correta da memória. Outra dificuldade foi em conseguir transformar o conteúdo extraído no formato JSON, uma vez que demoramos para entender a formatação correta.

3.3 Melhorias futuras

Algumas melhorias que poderiam ser feitas são:
→ Melhor formatação do texto dos comentários
→ Permitir ao utilizador dar o nome ao ficheiro JSON que está a criar;
O projeto foi realizado de acordo com o enunciado. As funcionalidades pedidas foram satisfeitas, com o esforço de todos os elementos do grupo. Conseguimos compreender a importância do uso de expressões regulares e, também melhorou nossa capacidade quanto à utilização destas novas ferramentas.

Apêndice A

Código do Programa

```
%{
/* Declaracoes C diversas */
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

typedef struct commentThread {
    char *id;
    char *userid;
    char *date;
    char *commentText;
    struct commentThread *prox;
    int count;
} *Comment;

Comment newcomment (int x) {
    Comment l = malloc (sizeof (struct commentThread ) );
    lprox = NULL;
    lcount = x;

    return l;
}

Comment inicio, *frente;

%}

%option stack

digit [0-9]
alphanum [0-9a-zA-Z]
id_formatt {alphanum}{8}-{alphanum}{4}-{alphanum}{4}-{alphanum}{4}-{alphanum}{12}
date_formatt {digit}{2}\.{digit}{2}\.{digit}{4}
```

```
com_section_init         li [' ']+class=\"comment\" [' ']+
com_section_fim         \<\ /li\>

com_list_init            \<ol [' ']+class=\"comments__list\"
com_list_fim            \<\ / ol\>

com_content_init      \<p\>['\n']*[' ']*(\t)*
com_content_fim      [' ']*(\t)*\<\/p\>

user_info_init           \/utilizador\/perfil\/
user_info_fim            \<\/a\>

date_info_init           \<a[' ']class=\"comment__permalink\">
date_info_fim            \<\/a\>

com_id_init              data-comment-id=
com_id_fim               \>

%s COMENTARIO    USER    TEXTO    ID_COMENTARIO    DATA

%%

{com_list_init}                    {  numero++;  }
{com_list_fim}                     {  numero--;  }
{com_section_init}                 {  yy_push_state(COMENTARIO); frente=&((*frente)prox); (*fren

<COMENTARIO>{com_id_init}               {  yy_push_state(ID_COMENTARIO);  }
<COMENTARIO>{user_info_init}       {  yy_push_state(USER);  }
<COMENTARIO>{date_info_init}       {  yy_push_state(DATA);  }
<COMENTARIO>{com_content_init}     {  yy_push_state(TEXTO); }
<COMENTARIO>{com_section_fim}      {  yy_pop_state();  }

<ID_COMENTARIO>{id_formatt}           {  (*frente)id =strdup( yytext);  }
<ID_COMENTARIO>{com_id_fim}           {  yy_pop_state(); }

<USER>{id_formatt}                    {  (*frente) userid =strdup( yytext); }
<USER>{user_info_fim}                {  yy_pop_state(); }

<DATA>{date_formatt}                 {  (*frente)date =strdup( yytext); }
<DATA>{date_info_fim}                {  yy_pop_state(); }

<TEXTO>[^\<]*                  {  (*frente)commentText =strdup( yytext); }
<TEXTO>{com_content_fim}       {  yy_pop_state(); }

(.|\n)                      {   ;  }

%%
```

```
int yywrap(){
  return(1);
}


int contarr(Comment x,int n){
  int r=0;
  x=x->prox;
  while( x!=NULL && x->count > n){
    r++;
    x=x->prox;
  }
  return r;
}

void espaco(FILE *f,int n){
  for(int i=1;i<n;i++) fprintf(f,"        ");
}


void displayc(FILE *f,char *a,int x){
  int i=4,j=strlen(a)-1;

  while( a[j]==' ' ||  a[j]=='\n' || a[j]=='\t') j--;
  while( a[i]==' ' ||  a[i]=='\n' || a[i]=='\t') i++;

  for(;a[i] && i<j  ;i++){
    if(a[i]=='\n') {fprintf(f,"\n");espaco(f,x);}
    else if(a[i]=='\"') fprintf(f,"\\\"");
    else fprintf(f,"%c",a[i]);
  }
}

void displayR(FILE *f,Comment s,int n){
  int i;

  espaco(f,s->count);
  fprintf(f,"\"id\": \"%s\",\n", s->id);

  espaco(f,s->count);
  fprintf(f,"\"user\": \"%s\",\n", s->userid);

  espaco(f,s->count);
  fprintf(f,"\"date\": \"%s\",\n", s->date);

  espaco(f,s->count);
  fprintf(f,"\"timestamp\": \"NA\",\n");
```

```
  espaco(f,s->count);
  fprintf(f,"\"commentText\": \"");
  displayc(f,s->commentText,s->count);
  fprintf(f,"\",\n");

  espaco(f,s->count);
  fprintf(f,"\"likes\": 0,\n");

  int x=contarr(s,s->count);

  espaco(f,s->count);
  fprintf(f,"\"hasreplies\": ");

  if(x==0) fprintf(f,"false,\n");
  else fprintf(f,"true,\n");

  espaco(f,s->count);
  fprintf(f,"\"numberOfReplies\":%d,\n",x);

  espaco(f,s->count);
  fprintf(f,"\"replies\": [");
  if(x==0) fprintf(f,"],");
  else fprintf(f,"  {");
  fprintf(f,"\n\n");
}

int main(){
  inicio=newcomment(numero);
  frente=&inicio;

  yylex();
  inicio=inicio->prox;


  frente=&inicio;
  FILE *f;
  f=fopen("bacon.json","w+");
  fprintf(f,"\n\"commentThread\": [\n");
  fprintf(f,"{\n");

  while(*frente!=NULL){

    displayR(f,*frente,(*frente)->count);

    if ((*frente)->prox!=NULL) {
      if((*frente)->count > (*frente)->prox->count) {
        fprintf(f,"\n");
```

```
        espaco(f,(*frente)->count);
        fprintf(f,"} ],\n");
          }
    }
    else {
      numero=(*frente)->count;
      while(numero > 1){
        fprintf(f,"\n");
        espaco(f,numero);
        fprintf(f,"}  ]\n");
        numero--;
         }
    }

    frente=&((*frente)->prox);

  }
  fprintf(f,"},\n];");
  fclose(f);

  return 0;
}
```

Apêndice B

Output do programa


```
"commentThread": [{
    "id": "541af870-e066-49f0-b08c-08d7a443991c",
    "user": "b4a7dc3b-5a7b-4a0a-85eb-539bcaddcafa",
    "date": "29.01.2020",
    "timestamp": "NA",
    "commentText": "Esta  sra Joacine  consegui enganar alguns; agora aguentem se a bronca. Vamos
    "likes": 0,
    "hasreplies": false,
    "numberOfReplies":0,
    "replies": [],

    "id": "fcbbab81-b314-42c9-e307-08d7a4444dbe",
    "user": "c7680446-9578-4617-9095-4d851f182e06",
    "date": "29.01.2020",
    "timestamp": "NA",
    "commentText": "Fiquei agora a saber que a Assunção Cristas é Angolana! Porque é que o Chega
    "likes": 0,
    "hasreplies": true,
    "numberOfReplies":4,
    "replies": [  {

            "id": "3beeb927-f969-4c48-fd69-08d7960b4eed",
            "user": "51e4df24-306a-444c-b319-09f5f5174ce2",
            "date": "29.01.2020",
            "timestamp": "NA",
            "commentText": "E a Ministra da Justiça também é angolana e o André também não pediu
            "likes": 0,
            "hasreplies": false,
            "numberOfReplies":0,
            "replies": [],

            "id": "99d53e73-9673-4f16-1e36-08d7a443f8a6",
            "user": "c7680446-9578-4617-9095-4d851f182e06",
            "date": "29.01.2020",
```

```
            "timestamp": "NA",
            "commentText": "Jorge, não faço ideia da razão. Se calhar ele ainda não sabe disso, t
            "likes": 0,
            "hasreplies": false,
            "numberOfReplies":0,
            "replies": [],

            "id": "e0326d03-37ab-4236-e319-08d7a4444dbe",
            "user": "43993747-edb9-4cfe-b23c-c38b08096cbe",
            "date": "29.01.2020",
            "timestamp": "NA",
            "commentText": "Porque a Cristas não tem ideias estúpidas... Pelo menos deste calibre
            "likes": 0,
            "hasreplies": false,
            "numberOfReplies":0,
            "replies": [],

            "id": "0956c3f0-2ad0-48e0-bfca-08d7a442b6a7",
            "user": "c7680446-9578-4617-9095-4d851f182e06",
            "date": "30.01.2020",
            "timestamp": "NA",
            "commentText": "Mas ideias estúpidas é o que não falta no Chega. O que devemos fazer
            "likes": 0,
            "hasreplies": false,
            "numberOfReplies":0,
            "replies": [],


            } ],
}],
```