Processamento de Linguagens e Compiladores LCC (3º ano)

Trabalho Prático Nº1 (FLex)

Ano lectivo 20/21

Grupo Nº9

Paulo Costa (A87986)

André Araújo (A87987)

16 de novembro de 2020

Resumo

Este trabalho prático tem como principais objectivos:

i) Aumentar a experiência de uso do ambiente Linux e de algumas ferramentas de apoio à programação;

ii) Aumentar a capacidade de escrever Expressões Regulares (ER) para descrição de padrões de frases;

iii) Desenvolver, a partir de ERs, sistemática e automaticamente Processadores de Linguagens Regulares, que filtrem ou transformem textos com base no conceito de regras de produção Condição-Ação;

iv) Utilizar o Flex para gerar filtros de texto em C.

Conteúdo

Capítulo 1

Introdução

Vamos então, começar por apresentar o respectivo enunciado. Em seguida, apresentar decicões que tomamos, assim como as decisões que lideraram o desenho da solução e a sua implementação (com a respetiva espeficação Flex).
Iremos ainda colocar alguns exemplos da sua utilização e uma breve conclusão deste primeiro trabalho de Processamento de Linguagens e Compiladores, em relação ao tema Flex.

1.1 Ex.3 - Transformador DailyExpress2NetLang

Concretize o enunciado genérico acima, considerando como base o ficheiro
$http : //www4.di.uminho.pt/\ prh/DailyExpress/DailyExpress_extraction_english_comments_14.html$ que contém os comentários a uma notícia publicada no jornal DailyExpress, o qual deve analisar com todo o cuidado. A ferramenta a desenvolver tem de funcionar bem para outros ficheiros recolhidos do mesmo jornal (na diretoria acima encontra mais exemplos para testar).

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

Para a resolução deste exercício tinhamos de realizar as seguintes tarefas:

1) Especificar os padrões de frases que quer encontrar no texto-fonte, através de ERs.

2) Identificar as acções semânticas a realizar como reacção ao reconhecimento de cada um desses padrões.

3) Identificar as Estruturas de Dados globais que possa eventualmente precisar para armazenar temporariamente a informação que vai extraindo do texto-fonte ou que vai construindo à medida que o processamento avança.

4) Desenvolver um Filtro de Texto para fazer o reconhecimento dos padrões identificados e proceder à transformação pretendida, com recurso ao Gerador FLex.

Tinhamos ainda, de ter em conta o facto de que estávamos a realizar algo que tinha como intuito, fazer um estudo sócio-linguístico da forma e conteúdo dos comentários que uma dada notícia de Jornal suscitou.
Estes dados, relevantes à análise pretendida, tinham de ser retirados automaticamente de cada ficheiro HTML extraído da versão online desse Jornal, devendo depois ser transformados no formato JSON a seguir mostrado.

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

Com isto, tinhamos de construir então, um filtro de texto, recorrendo ao gerador Flex, que realize o processamento explicado e seguindo evidentemente o mesmo formato apresentado.

2.2 Especificação do Requisitos

2.2.1 Dados

Primeiramente, começamos por tentar identificar indicadores, no código html, que nos tornassem possível indicar o início e o fim da informação a preencher, para cada um dos campos da estrutura.
Colocamos, agora um esquema que fizemos logo no início da resolução do trabalho, que marcava os indicadores que teriamos de usar, do código html, que nos iriam a ser úteis.

```
1-Nome
2-Data
3-Comentário
4-Likes
5-Replies
6-Destinatário

<li aria-label="Comment" class="spcv_list-item"><div class="spcv_safari-padder"></div>
1.....data-spot-im-class="message-username" tabindex="27">GMJay</span>
2.....data-spot-im-class="message-timestamp">14Â Aug</time>
3.....data-spot-im-class="message-text">I agree with him. "In a time of universal deceit,
telling the truth is a revolutionary act"</div>
4.....class="spcv_number-of-votes">103 Likes</span>
5.....<li aria-label="Comment reply" role="article"><div class="spcv_message-stack">
<div class="spcv_appearance-component spcv_is-reply"><div class="spcv_message-view spcv_depth-1
spcv_conversation-message" data-message-depth="1"
data-message-id="sp_9LMINbK9_1165460_c_Kvicz0_r_SBC2yz" data-spot-im-class="message-view">
<div class="spcv_marked-component"><div class="spcv_marker"></div>
5.1.....data-spot-im-class="message-username" tabindex="41">cynicalme</span>
2.1.....data-spot-im-class="message-timestamp">14Â Aug</time>
6.......<span class="spcv_username">GMJay</span>
3.1.....data-spot-im-class="message-text">Yes, you beat me to it.</div>
4.1.....class="spcv_number-of-votes">17 Likes</span>
5.3.....<li aria-label="Comment reply" role="article"><div class="spcv_message-stack">
<div class="spcv_appearance-component spcv_is-reply"><div class="spcv_message-view spcv_depth-2
spcv_conversation-message" data-message-depth="2"
data-message-id="sp_9LMINbK9_1165460_c_Kvicz0_r_FYSyKp" data-spot-im-class="message-view">
<div class="spcv_marked-component"><div class="spcv_marker"></div>
5.4.....data-spot-im-class="message-username" tabindex="48">thesageandonion</span>
2.2.....data-spot-im-class="message-timestamp">14Â Aug</time>
6.1.....<span class="spcv_username">cynicalme</span>
3.2.....data-spot-im-class="message-text">spot  on</div>
4.2.....class="spcv_number-of-votes">11 Likes</span>
```

2.2.2 Pedidos

Bem, como pedidos, tinhamos que colocar todas as informações obtidas apartir do código html, num ficheiro (.json) onde eram precisas todas as informações de forma a conseguirmos preencher os campos para completar a estrutura pretendida, que mais uma vez, volto a apresentar:

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

2.2.3 Relações

Como relações, entre os dados e os pedidos, temos que: para obter todos os dados tivemos de delinear/definir expressões regulares que nos tornassem possível obter as infromações pretendidas e em seguida, preencher os campos da estrutura. Isto, porque tanto para obter os dados, como para preencher a estrutura pretendida, precisamos de expressões regulares que nos permitam fazer isso.
Na secção "Algoritmos", iremos apresentar 2 exemplos de expressões regulares que utilizamos.

Capítulo 3

Concepção/desenho da Resolução

3.1 Estruturas de Dados

Aqui, começamos por mostrar as biblioteas que utilizamos no código, seguidas de um buffer auxiliar para guardar temporariamente as strings.
Depois temos a estrutura que alverga todos os compunentes de cada comentario, ou seja, os apontadores para os ficheiros pretendidos.
Acabamos com apontadores para a estrutura auxiliar (inicio e next), um array que guarda os apontadores da estrutura para guardar vários replies, um indicador, seguido de uns auxiliares e, por último, mas não menos importante, um inteiro que nos permite gerar um id aleatório para as mensagens, que no caso não o continham.

```
%{
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
char buf[2000];
typedef struct commentThread{
        char* id;
        char* user;
        char* date;
        char* commentText;
        int nlikes;
        int nReplies;
        struct commentThread* replies;
        struct commentThread* next;
}*CommentThread;
extern FILE *yyin, *yyout;
CommentThread inicio, next;
CommentThread a[20];
int i=-1;
int depth,aux;
int id=200;
%}
```

3.2 Algoritmos

Aqui conseguimos acder ás funções de stack: $\{yy\_push\_state();\}$ e $\{yy\_pop\_state();\}$, que nos são necessárias no código, que em seguida vai ser também apresentado.

```
%option stack
```

Agora, decidimos perante as "tags"que encontramos, para associar onde devíamos ir buscar cada um dos parâmetros, delinear as Expressões Regulres (ER) que nos permitiram conseguir obter as formações pretendidas.
Apresentamos, em seguida, o exemplo da ER, utilizada para obter o campo "user"e ainda para o campo "data", as restantes foram realizadas de modo análogo:

```
%x HID HUSER HDATE htimestamp HCOMMENT HLIKES
%s HREPLIES

%%

||user||

data\-spot\-im\-class\=\"message-username\"" "tabindex\=\"[0-9]*\"\>
                                    {yy_push_state(HUSER);}
<HUSER>\<\/span>                    {yy_pop_state();}
<HUSER>[^<]+                        {a[(20+i)%20]->user=strdup(yytext);}

||data||


"data-spot-im-class="\""message-timestamp"\"\>
                                    {yy_push_state(HDATE);}
<HDATE>\<\/time\>                   {a[(20+i)%20]->date=strdup(buf);buf[0]='\0';yy_pop_state();}
<HDATE>                        {;}
<HDATE>[^(<|;|&)]+                  {strcat(buf,yytext);strcat(buf," \0");}

(...)

%%
```

Colocámos aqui, ainda como Algoritmo, esta função auxiliar, que neste caso, coloca num ficheiro as informações recolhidas no html e faz com que o "output"apareça da forma pretendida, transformado-as no formato JSON, apresentado anteriormente.

```
void writejson(FILE* yyout,CommentThread s)
{
        while (s!=NULL)
        {
                fprintf(yyout,"{\n");
                if(s->id==NULL){
                        fprintf(yyout,"\"id\": \"%d\",\n",id++);
                }else{
                        fprintf(yyout,"\"id\": \"%s\",\n",s->id);
                }
                fprintf(yyout,"\"user\": \"%s\",\n",s->user);
                fprintf(yyout,"\"date\": \"%s\",\n",s->date);
                fprintf(yyout,"\"timestamp\": \"NA\",\n");
                fprintf(yyout,"\"commentText\": \"%s\",\n",s->commentText);
                fprintf(yyout,"\"likes\": %d,\n",s->nlikes);
                if (s->nReplies==0){
                        fprintf(yyout,"\"hasReplies\": false,\n");
                        fprintf(yyout,"\"numberOfReplies\": 0,\n");
                        fprintf(yyout,"\"replies\": []\n");
                }else{
                        fprintf(yyout,"\"hasReplies\": true,\n");
                        fprintf(yyout,"\"numberOfReplies\": %d,\n",s->nReplies);
                        fprintf(yyout,"\"replies\": [\n");
                        writejson(yyout,s->replies);
                        fprintf(yyout,"]\n");
                }
                if (s->next==NULL){
                        fprintf(yyout,"}\n");
                }else{
                    fprintf(yyout,"},\n");
                }
                next=s;
                s=s->next;
                free(next);
        }
}
```

E aqui colocámos também a função main, aquela que tem as funções *yyin* e *out*, que permite abrir o ficheiro em modo leitura e em modo de escrita, respetivamente. E aquela que chama todas as funções necessárias para o processamento e a concretização do exercício realizado!

```
int main(int argc,char* argv[])
{
        buf[0]='\0';
        if (argc>1){strcpy(buf,argv[1]);}
        else{strcpy(buf,"exp.html");}
        yyin=fopen(buf,"r");


        inicio=malloc(sizeof(struct commentThread));
        buf[0]='\0';
        inicio->nlikes=0;
        a[20-1%20]=inicio;

        yylex();
        buf[0]='\0';
        if (argc>2){strcpy(buf,argv[2]);}
        else{strcpy(buf,"out.json");}
        yyout=fopen(buf,"w");
        fprintf(yyout,"{\n\"commentThread\": [\n");
        writejson(yyout,inicio->replies);
        free(inicio);
        fprintf(yyout,"]\n}\n");

        return 0;
}
```

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

O maior problema que este exercício nos deu, talvez no sentido em que foi a altura do trabalho em que mais tentativas e ajustes tivemos foi na expressão regular relativa aos replies.
Passo agora a apresentá-la;

```
data\-message\-depth\=\"          {yy_push_state(HREPLIES);}
<HREPLIES>\"                      {yy_pop_state();}
<HREPLIES>[0-9]+       {
                         depth=atoi(yytext);
                         next=malloc(sizeof(struct commentThread));
                         next->nlikes=0;
                         next->nReplies=0;
                         if (depth<=i){
                              if(depth<i){
                                   for(;depth!=i;i--){
                                        a[(20+i-1)%20]->nReplies+=a[(20+i)%20]->nReplies;
                                        }
                                   a[(20+i-1)%20]->nReplies+=a[(20+i)%20]->nReplies;
                              }

                             a[(20+i)%20]->next=next;
                             a[(20+i)%20]=a[(20+i)%20]->next;
                         }else{
                             if(depth==(i+1)){
                             a[(20+i)%20]->replies=next;
                             a[(i+1)%20]=a[(20+i)%20]->replies;
                              i++;

                         }

                         }
                         a[(20+i-1)%20]->nReplies+=1;
                   }
```

Achamos que possa ter sido mais complicada, porque mexer com a posição da stack, ou seja, estávamos com dificuldade em ter em conta as posições corretas dos apontadores da estrutura.
Algo que acabou por ficar resolvido, quando começamos a trabalhar com o resto da divisão inteira por 20, aí foi quando realmente conseguimos "controlar"as posições onde nos encontravamos e portanto, acabou por facilitar a resolução do trabalho até ao seu final.

4.2 Testes realizados e Resultados

Primeiro, mostraremos o resultado que obtemos apartir de um html, onde teriámos um comentário "principal", com dois replies.
Colocamos aqui parte do código html, que contém informações que nos são úteis, para o processamento do problema, tal como referido na secção 2.2 .

```
(...) data-message-depth="0" (...)
(...) data-spot-im-class="message-username" tabindex="6">Cooperj</span>
(...) data-spot-im-class="message-timestamp">22Â Aug</time>
(...) data-spot-im-class="message-text">Who cares really?</div>
(...) span class="spcv_number-of-votes">34 Likes</span>

(...) data-message-depth="1" (...)
(...) data-spot-im-class="message-username" tabindex="13">SRFexport</span>
(...) data-spot-im-class="message-timestamp">23Â Aug</time>
(...)<span class="spcv_username">Cooperj</span>
(...) data-spot-im-class="message-text">The express, and a lot</div>

(...) data-message-depth="1" (...)
(...) data-spot-im-class="message-username" tabindex="20">Hotlegs-Gigi</span>
(...) data-spot-im-class="message-timestamp">22Â Aug</time>
(...) <span class="spcv_username">Cooperj</span>
(...) data-spot-im-class="message-text">Um, no one !</div>
(...) span class="spcv_number-of-votes">6 Likes</span>
```

Depois de termos o código html, guardado no nosso computador, com o nome (”exp.html”), basta irmos à consola e efetuarmos os seguintes comandos:

```
pc@pc-VirtualBox:~$ cd Desktop
pc@pc-VirtualBox:~/Desktop$ cd Trab1
pc@pc-VirtualBox:~/Desktop/Trab1$ flex trabalho.l
pc@pc-VirtualBox:~/Desktop/Trab1$ gcc -o trabalho lex.yy.c
pc@pc-VirtualBox:~/Desktop/Trab1$ ./trabalho "exp.html" "out.json"
```

Por fim, neste curto exemplo, mas muito demonstrativo, obteríamos o ficheiro ”out.json”, com o seguinte conteúdo:


```
{
"commentThread": [
{
"id": "30",
"user": "Cooperj",
"date": "22Â Aug ",
"timestamp": "NA",
"commentText": "Who cares really?",
"likes": 34,
"hasReplies": true,
"numberOfReplies": 2,
"replies": [
{
"id": "200",
"user": "SRFexport",
"date": "23Â Aug ",
"timestamp": "NA",
"commentText": "The express, and a lot",
"likes": 0,
"hasReplies": false,
"numberOfReplies": 0,
"replies": []
},
{
"id": "201",
"user": "Hotlegs-Gigi",
"date": "22Â Aug ",
"timestamp": "NA",
"commentText": "Um, no one !",
"likes": 6,
"hasReplies": false,
"numberOfReplies": 0,
"replies": []
}
]
}
```

Como mais um exemplo, queremos mostrar apenas como ficaria o output do exercício, mas tendo apenas o código html fornecido, dois comentários, ambos "principais".

```
(...) data-message-depth="0" (...)
(...) data-spot-im-class="message-username" tabindex="27">ChampagneDosser</span>
(...) data-spot-im-class="message-timestamp">22Â Aug</time>
(...) data-spot-im-class="message-text">Men like Chris Evans and Bojo all get to date young
women while looking like obese scarecrows so why does she need to explain herself?</div>
(...) span class="spcv_number-of-votes">17 Likes</span>

(...) data-message-depth="0" (...)
(...) data-spot-im-class="message-username" tabindex="34">Hotlegs-Gigi</span>
(...) data-spot-im-class="message-timestamp">22Â Aug</time>
(...) data-spot-im-class="message-text">She really doesn't do herself any favours does she.</div>
(...) span class="spcv_number-of-votes">14 Likes</span>
```

Realizando os comandos referidos no exemplo anterior, na consola, acabamos por obter o ficheiro "out.json", com o seguinte conteúdo:


```
{
"id": "29",
"user": "ChampagneDosser",
"date": "22Â Aug ",
"timestamp": "NA",
"commentText": "Men like Chris Evans and Bojo all get to date young women while looking like
obese scarecrows so why does she need to explain herself?",
"likes": 17,
"hasReplies": false,
"numberOfReplies": 0,
"replies": []
},
{
"id": "28",
"user": "Hotlegs-Gigi",
"date": "22Â Aug ",
"timestamp": "NA",
"commentText": "She really doesn't do herself any favours does she.",
"likes": 14,
"hasReplies": false,
"numberOfReplies": 0,
"replies": []
}
```


Gostaríamos de apresentar mais exemplos, e de maior complexidade, mas também achamos que o documento acabava por ficar demasiado exaustivo e também achamos que com estes exemplos, conseguímos demonstrar que de facto, acabamos por conseguir o objetivo pretendido.

Capítulo 5

Conclusão

Bem, em estilo de conclusão, achamos que este trabalho, não foi fácil, sobretudo na parte inicial, ou seja, na parte em que nos tínhamos de enquadrar com o objetivo do problema, e que tínhamos de delinear estratégias para o resolvermos e ainda na parte de trabalhar com os replies dos comentários.

Mas, no geral, sem dúvida alguma que este exercício é muito útil, porque torna a pagína do jornal Daily Express, legível, sem todos aqueles espacços em branco, e ao analisar este documento tudo se torna mais fácil, e o tempo que se acaba por poupar é enorme.

Achamos ainda que este trabalho deu para consolidarmos a matéria sobre o Flex, porque sem dúvida que a prática acaba por ser a melhor maneira de perceber/entender a parte teórica.
E depois de termos realizado este trabalho, sentimos mesmo que o produto final foi algo muito positivo, porque conseguimos alcançar todos os objetivos pretendidos e sobretudo foi interessante a realização do mesmo!

Com isto, esperemos que o trabalho esteja do agrado do professor e que tire proveito do mesmo, porque isso seria um excelente sinal!

Na próxima folha, está apresentado o Código final do Programa.

Obrigado pela atenção,
Cumprimentos!

Apêndice A

Código do Programa

```
%{
/* Declaracoes C diversas */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
char buf[2000];
typedef struct commentThread{
        char* id;
        char* user;
        char* date;
        char* commentText;
        int nlikes;
        int nReplies;
        struct commentThread* replies;
        struct commentThread* next;
}*CommentThread;

extern FILE *yyin, *yyout;
CommentThread inicio, next;
CommentThread a[20];
int i=-1;
int depth,aux;
int id=200;
%}


%option stack

%x HID  USERT HUSER HDATE htimestamp HCOMMENT HLIKES
%x HREPLIES
```

```
%%
data\-message\-depth\=\"          {yy_push_state(HREPLIES);}
<HREPLIES>\"                      {yy_pop_state();}
<HREPLIES>[0-9]+       {
                            depth=atoi(yytext);
                            next=malloc(sizeof(struct commentThread));
                            next->nlikes=0;
                            next->nReplies=0;
                            if (depth<=i){
                                 if(depth<i){
                                      for(;depth!=i;i--){
                                           a[(20+i-1)%20]->nReplies+=a[(20+i)%20]->nReplies;
                                           }
                                         a[(20+i-1)%20]->nReplies+=a[(20+i)%20]->nReplies;
                                 }

                                a[(20+i)%20]->next=next;
                                a[(20+i)%20]=a[(20+i)%20]->next;
                            }else{
                                if(depth==(i+1)){
                                a[(20+i)%20]->replies=next;
                                a[(i+1)%20]=a[(20+i)%20]->replies;
                                 i++;

                            }

                            }
                            a[(20+i-1)%20]->nReplies+=1;
                     }

data\-spot\-im\-class\=\"message-username\"" "tabindex\=\"[0-9]*\"\>
                             {yy_push_state(HUSER);}
<HUSER>\<\/span>             {yy_pop_state();}
<HUSER>[^<]+                 {a[(20+i)%20]->user=strdup(yytext);}


"data-spot-im-class="\""message-timestamp"\"\>
                             {yy_push_state(HDATE);}
<HDATE>\<\/time\>            {a[(20+i)%20]->date=strdup(buf);buf[0]='\0';yy_pop_state();}
<HDATE>                 {;}
<HDATE>[^(<|;|&)]+           {strcat(buf,yytext);strcat(buf," \0");}
```

```
data\-spot\-im\-class\=\"message-text\"\>(\<ul" "class\=\"spcv_ul\"\>)?
                                    {yy_push_state(HCOMMENT);}
class\=\"spcv\_is\-deleted\"\>\<span\>
                                    {yy_push_state(HCOMMENT);}
<HCOMMENT>(\<\/div\>)
                               {a[(20+i)%20]->commentText=strdup(buf);buf[0]='\0';yy_pop_state();}
<HCOMMENT>(\<\/?strong\>|\<\/?span[^>]*\>|\<\/?li\>|\<\/ul\>|\<\/?em\>)
                                    {;}
<HCOMMENT>\"
                                    {strcat(buf,"\\\"\0");} \\substitui as "" por \"
<HCOMMENT>[^(<|\n|\")]+
                                    {strcat(buf,yytext);}

class\=\"spcv\_number\-of\-votes\"\>
                                    {yy_push_state(HLIKES);}
<HLIKES>Likes?\<\/span\>
                                    {yy_pop_state();}
<HLIKES>[0-9]+
                                    {a[(20+i)%20]->nlikes=atoi(yytext);}

id\=\"comment\-description\-         {yy_push_state(HID);}
<HID>\-?[0-9]+                      {a[(20+i)%20]->id=strdup(yytext);}
<HID>\"                             {yy_pop_state();}

<*>(.|\n)                           {;}

%%


int yywrap()
{ return(1); }

void writejson(FILE* yyout,CommentThread s)
{
        while (s!=NULL)
        {
                fprintf(yyout,"{\n");
                if(s->id==NULL){
                        fprintf(yyout,"\"id\": \"%d\",\n",id++);
                }else{
                        fprintf(yyout,"\"id\": \"%s\",\n",s->id);
                }
                fprintf(yyout,"\"user\": \"%s\",\n",s->user);
                fprintf(yyout,"\"date\": \"%s\",\n",s->date);
                fprintf(yyout,"\"timestamp\": \"NA\",\n");
                fprintf(yyout,"\"commentText\": \"%s\",\n",s->commentText);
                fprintf(yyout,"\"likes\": %d,\n",s->nlikes);
```

```
                if (s->nReplies==0){
                        fprintf(yyout,"\"hasReplies\": false,\n");
                        fprintf(yyout,"\"numberOfReplies\": 0,\n");
                        fprintf(yyout,"\"replies\": []\n");
                }else{
                        fprintf(yyout,"\"hasReplies\": true,\n");
                        fprintf(yyout,"\"numberOfReplies\": %d,\n",s->nReplies);
                        fprintf(yyout,"\"replies\": [\n");
                        writejson(yyout,s->replies);
                        fprintf(yyout,"]\n");
                }
                if (s->next==NULL){
                        fprintf(yyout,"}\n");
                }else{
                    fprintf(yyout,"},\n");
                }
                next=s;
                s=s->next;
                free(next);
        }
}

int main(int argc,char* argv[])
{
        buf[0]='\0';
        if (argc>1){strcpy(buf,argv[1]);}
        else{strcpy(buf,"exp.html");}
        yyin=fopen(buf,"r");


        inicio=malloc(sizeof(struct commentThread));
        buf[0]='\0';
        inicio->nlikes=0;
        a[20-1%20]=inicio;

        yylex();
        buf[0]='\0';
        if (argc>2){strcpy(buf,argv[2]);}
        else{strcpy(buf,"out.json");}
        yyout=fopen(buf,"w");
        fprintf(yyout,"{\n\"commentThread\": [\n");
        writejson(yyout,inicio->replies);
        free(inicio);
        fprintf(yyout,"]\n}\n");

        return 0;
}
```