Modelos de Computación (2015-2016)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 4

José Carlos Martínez Velázquez

10 de noviembre de 2015

1. Crear una plantilla en *Lex* para encontrar

Antes de comenzar a explicar cómo he construído el ejercicio, me gustaría comentar que he escrito un programa lex para cada apartado y posteriormente he aunado todo lo que se pide en un sólo programa.

Las diferentes partes de un programa lex son delimitadas con '%%', de tal manera que, desde el principio hasta el primer delimitador se denomina sección de declaraciones, donde se añaden bibliotecas de C o se definen macros, delimitado con '{% [...]%}', podremos asignar alias a determinadas expresiones regulares y también se pueden declarar las cabeceras de las funciones que desarrollaremos en la última sección.

El segundo bloque es la sección de reglas, donde asignaremos código C cuando se encuentre una expresión que coincide con la expresión regular en cuestión.

Por último, la tercera división es la sección de código C, que nos permite desarrollar el main(), redefinir funciones que ya defina Lex o definir las que se declararon como cabeceras en la sección de declaraciones.

1. **Direcciones de correo electrónico y de páginas web en un fichero texto.**

```
1  %{
2  #include <stdio.h>
3  %}
4  numero   [0-9]
5  dominio  [a-z]+\.*
6  term          \.[a-z]{2,3}
7  correo   .+@{dominio}+{term}
8  pagina   "http://"?"www."?.+"."{dominio}*{term}("/".*)*
9
10
11 %%
12 {correo}          {printf("Encontrado un correo: %s\n", yytext);}
13
14 {pagina}          {printf("Encontrada una pagina: %s\n", yytext);}
15 .                         {}
16
17 %%
18 int main(int argc, char* argv[]){
19      if(argc==2){
20          yyin=fopen(argv[1],"rt");
21          if(yyin==NULL){
22                  printf("El fichero %s no se puede abrir\n",argv[1]);
23                  exit(-1);
24          }
25      }else{yyin=stdin;}
26      yylex();
27      return 0;
28 }
```

Dado que sólo vamos a imprimir caracteres en éste primer programa, la única biblioteca de C que vamos a necesitar es la que nos permite hacer printf(): <stdio.h>.

Vamos a definir también aquéllas expresiones regulares que vamos a utilizar en la siguiente sección. Estas son:

- ***numero:*** Cualquier dígito del 1 al 9 (sólo uno).
- ***dominio:*** Una sucesión de uno o más caracteres seguidos por un punto que puede aparecer o no: gmail[.] hotmail[.] ugr[.] etc.
- ***terminacion (term):*** Es una sucesión de 2 o 3 caracteres precedidos por un punto .ugr, .correo, etc. El punto debe estar porque si el correo sólo tiene terminación, ésta siempre debe ir precedida por un punto. Veamos un ejemplo: xxxxx@[correo.]*(dominio1)*[ugr]*(último dominio)*.es*(la terminación debe contener un punto para ser válida, de hecho, lo es).*
- ***correo:*** Es una sucesión de caracteres (al menos uno) seguido de una y solo una '@', seguida de uno o más dominios (al menos debe tener @gmail, @hotmail, @correo, @ugr, etc) y una sola terminación: .com, .es, etc.

Con ésto ya podríamos reconocer un correo electrónico. Ahora vamos a ver cómo reconocer una página web. Para ello definimos pagina, que se compondrá del literal "http://", que puede aparecer sólo una vez o ninguna, seguido del literal "www." que también puede aparecer o no. Posteriormente cualquier caracter, que componga la dirección en sí y luego un caracter "." que nos lleve a definir la terminación de la página. Nos aprovecharemos de dominio y terminación, anteriormente definidos. Una página web puede tener varios dominios pero siempre una única terminación: `decsai.ugr.es/...` `loquesea.blogs.edu.com/...` Además para separar los recursos, después de la terminación debe llevar una barra seguida de cualquier caracter y éste conjunto puede aparecer 0 (acceso a la página principal) o n veces (profundidad n en el árbol de directorios).

2. **Fechas, Números de teléfono, Matrículas, Códigos postales y NIF en otro fichero.**

```
1  %{
2  #include <stdio.h>
3  int cFechaNormal=0;
4  int cFechaIng=0;
5  int cTlf=0;
6  int cMat=0;
7  int cCP=0;
8  int cDni=0;
9  void resumen();
10 %}
11 separador     ("/"|"-")
12 dia           (0?[1-9]|[10-30]|31)
13 mes           (0?[1-9]|[10-12])
14 anio          ([0-9]{2}|[0-9]{4,})
15 fechaNormal   {dia}{separador}{mes}{separador}{anio}
16 fechaInglesa  {anio}{separador}{mes}{separador}{dia}
17 pre_int       \+[0-9]{2,3}
```

```
18 movil          (6|7)[0-9]{8}
19 fijo           (8|9)[0-9]{8}
20 telefono       {pre_int}+({fijo}|{movil})
21 matricula_t1 [0-9]{4}"-"?([B-D]|[F-H]|[J-N]|[P-T]|[V-Z]){3}
22 matricula_t2 [A-Z]{2}"-"?[0-9]{4}"-"?[A-Z]{2}
23 dni            [X-Z]{1}-?[0-9]{7}-?[a-zA-Z]|[0-9]{8}-?[a-zA-Z]
24 cod_post          (0[1-9]|[1-4][0-9]|5[0-2])[0-9]{3}
25
26 %%
27
28 {fechaNormal}    {cFechaNormal+=1;printf("Encontrada una fecha en
       formato dd/mm/aaaa: %s\n", yytext); REJECT;}
29 {fechaInglesa}  {cFechaIng+=1; printf("Encontrada una fecha en formato
       aaaa/mm/dd: %s\n", yytext);}
30 {telefono}                {cTlf+=1; printf("Encontrado un telefono: %s\n
       ", yytext);}
31 {matricula_t1}  {cMat+=1; printf("Encontrada una matricula: %s\n",
       yytext);}
32 {matricula_t2}  {cMat+=1; printf("Encontrada una matricula: %s\n",
       yytext);}
33 {cod_post}               {cCP+=1; printf("Encontrado un codigo postal:
       %s\n", yytext);}
34 {dni}                    {cDni+=1; printf("Encontrado un dni: %s\n",
       yytext);}
35 .                                {}
36
37 %%
38
39 int main(int argc, char* argv[]){
40     if(argc==2){
41         yyin=fopen(argv[1],"rt");
42         if(yyin==NULL){
43                 printf("El fichero %s no se puede abrir\n",argv[1]);
44                 exit(-1);
45         }
46     }else{yyin=stdin;}
47     yylex();
48     resumen();
49     return 0;
50 }
51 void resumen(){
52         printf("Se han encontrado %i fechas en formato dd/mm/aa.\n",
               cFechaNormal);
53         printf("Se han encontrado %i fechas en formato aaaa/mm/dd.\n",
                cFechaIng);
54         printf("Se han encontrado %i numeros de telefono.\n", cTlf);
55         printf("Se han encontrado %i matriculas.\n", cMat);
56         printf("Se han encontrado %i codigos postales.\n", cCP);
57         printf("Se han encontrado %i dni.\n", cDni);
58 }
```

Algunas de las cosas que aquí se piden son complicadas de representar, si se quiere ser muy preciso y fiel a la realidad. Por ejemplo, que no exista el día 30 de febrero

es complicado y en algún sitio hay que cortar, por eso nos haremos una idea de cómo puede ser representada una fecha, aunque en éste caso tendrá ése déficit. En éste apartado iremos definiendo lo que se ha hecho para cada cosa que se pide.

En primer lugar se ha declarado una variable contador para cada tipo de dato, que se incrementará en cada caso. También he definido una cabecera para una función resumen() que dirá por pantalla cuántas ocurrencias hay de cada tipo usando las variables contador.

- ***Fechas:*** Quizás la más dificil. He definido el alias separador, que separa dia, mes y año. Puede ser del tipo "-" ó "/", aunque se podrían definir más. Por otro lado, he definido los dias. Para que sean de dos dígitos, he hecho la unión de los lenguajes que van precedidos por 0, del 10 al 30 y el 31: 01,02,...,09,10,...30,31. Para definir los meses he seguido una estrategia similar. Para los años he realizado la unión de los lenguajes que representan los años con dos dígitos o con 4 o más (¿por qué quedarse en el año 9999?).

  He querido distinguir entre dos formatos de fecha, la que usamos nosotros dd(/-)mm(/-)aaaa ó dd(/-)mm(/-)aa y la fecha anglosajona, americana... aaaa(/-)mm(/-)dd ó aa(/-)mm(/-)dd. Cabe destacar que cuando el año se representa con dos dígitos, la fecha en cuestión pertenece a los dos tipos.

- ***Numeros de teléfono:*** Un numero de teléfono se compone del un prefijo internacional y un numero de fijo o móvil. Ambos deben cumplir ciertas restricciones. El prefijo internacional se compone del caracter "+" seguido de dos o tres dígitos, puede aparecer o no. El número que viene a continuación del prefijo internacional puede ser fijo o móvil. Si es fijo, puede empezar por 9 ó por 8 (ha empezado a usarse recientemente) y si es móvil, debe empezar por 6 ó 7 (que también se ha empezado a usar recientemente). Como un número de teléfono sin prefijo internacional se compone de 9 números y ya hemos definido el primero, lo que sige al primer dígito son 8 números del 0 al 9.

- ***Matrículas:*** Principalmente hay dos tipos. Las de antes de 2000 y las actuales. Las actuales, las que denominaremos matricula_t1 están compuestas de 4 dígitos del 0 al 9 seguidas de tres letras que no incluyen vocales. Hay quien las representa con un guión entre los números y las letras, por eso debe aparecer o no.

  Las anteriores, que denotaremos como matricula_t2, se componen de dos letras (o una), que representan la provincia (para no representar todas las provincias, aceptaremos cualesquiera dos letras al principio), 4 dígitos del 0 al 9 y otras dos letras al final, entre las que se aceptan vocales. Al igual que antes, se puede introducir un guión entre los números y las letras.

- ***Códigos postales:*** Un código postal (al menos en España) se compone de un número del 01 al 52, seguido de otros 3 números cada uno del 0 al 9. Siguiendo una estrategia similar a la de los días y los meses de la fecha, he restringido esos dos digitos a la union de lenguajes 01,...,09,10,...,19,20...,29,30,...,39,40,...,49,50,51,52. Del 10 al 49 se hace cogiendo un sólo número entre el 0 y el 4 y otro número

entre el 0 y el 9. Dado que no existe el 00, éste lenguaje debe ser distinto a 01,...,09 y dado que los que empiezan por 50 no pueden llegar al 9 (ni siquiera al 3), debe ser otro lenguaje diferente, uniendo los lenguajes se consiguen todos los codigos postales Españoles.

- ***DNI:*** Los DNI pueden ser de gente con permiso de residencia o no. Un DNI de un ciudadano con permiso de residencia debe empezar por la letra X, Y ó Z, seguida de 7 números y una letra. Un DNI de un ciudadano español se compone de 8 dígitos del 0 al 9 y una sola letra. Para simplificar las cosas, aceptaremos cualquier combinación de éstas, ya que la letra se calcula a partir del número de DNI. Como los dos tipos tienen formatos distintos, he recurrido a la unión de los dos lenguajes para poder cubrir todos los DNI de ciudadanos de España.

3. **Comentarios en un fichero con código fuente C++.**

```
1  %{
2  #include <stdio.h>
3  #include <string.h>
4  int cMl=0;
5  int cSl=0;
6  void resumen();
7  %}
8
9  apertura_ml      "/\*"
10 cerradura_ml     "\*/"
11 comentario       [^/\*]
12 apertura_sl      "//"
13 ml_comment       {apertura_ml}{comentario}*{cerradura_ml}
14 sl_comment       {apertura_sl}.*
15 %%
16
17 {ml_comment}     {cMl+=1;printf("Encontrado un comentario multilinea: %
       s\n", yytext);}
18 {sl_comment}     {cSl+=1;printf("Encontrado un comentario monolinea: %s
       \n", yytext);}
19 .                                {}
20
21 %%
22
23 int main(int argc, char* argv[]){
24     if(argc==2){
25         yyin=fopen(argv[1],"rt");
26         if(yyin==NULL){
27                 printf("El fichero %s no se puede abrir\n",argv[1]);
28                 exit(-1);
29         }
30     }else{yyin=stdin;}
31     yylex();
32     resumen();
33     return 0;
```

```
34 }
35 void resumen(){
36     printf("Se han encontrado %i comentarios multilinea.\n", cMl);
37     printf("Se han encontrado %i comentarios monolinea.\n", cSl);
38 }
```

En un fichero de C++ podemos distinguir dos tipos de comentarios: los monolínea y los multilínea. Los monolínea se abren con "//" y no tienen etiqueta de cierre. Se puede detectar que un comentario monolínea ha acabado cuando nos encontramos con un salto de línea después de la doble barra. Los multilínea en cambio tienen etiqueta de apertura "/*" y etiqueta de cierre "*/", ya que pueden contener un salto de línea sin dejar de ser comentario. En el programa vamos a definir una variable contador para cada tipo de comentario que utilizaremos posteriormente en la función resumen(), que nos dirá cuántos comentarios de cada tipo se han encontrado. Para detectar los distintos comentarios en un fichero C++ definimos las siguientes reglas:

- ***apertura_ml:*** Es la apertura de un comentario multilínea. El literal "/*".
- ***cerradura_ml:*** Es la cerradura de un comentario multilínea. El literal "*/".
- ***comentario:*** Un comentario multilínea puede componerse de todo (saltos de línea, tabuladores, en definitiva cualquier caracter, pero cualquier caracter tambien incluye "*" y "/", es decir, la etiqueta de cierre. Para definir un comentario correctamente, diremos que es todo lo que no sea "*/", así cuando encuentre la etiqueta de cierre, sabrá que se trata del fin del comentario. Cabe destacar que el comentario multilínea así definido permite que se encuentren los caracteres "*" y "/" pero por separado, es decir, con al menos un caracter distinto entre ellos.
- ***apertura_sl:*** Es la apertura de un comentario monolínea. El literal "//".
- ***ml_comment:*** Un comentario multilínea se va a componer de una etiqueta de apertura multilínea (apertura_ml), cualquier cosa que no sea la etiqueta de cierre (comentario) y una etiqueta de cierre (cerradura_ml).
- ***sl_comment:*** Un comentario monolínea se va a componer de una etiqueta de apertura "//" y cualquier caracter. Dado que la regla '.' (cualquier caracter) no incluye saltos de línea, en cuanto se encuentre uno sabrá que se ha acabado el comentario monolínea.

Para terminar, he aunado los tres programas que se pedían en uno solo. La definicion de reglas es tal cual está en cada fichero, pero reunidas. Lo único que cambia es el main(), que permite pasar varios archivos por parámetro.

```
1  %{
2  #include <stdio.h>
3  #include <string.h>
4  int cFechaNormal=0;
5  int cFechaIng=0;
6  int cTlf=0;
7  int cMat=0;
8  int cCP=0;
9  int cDni=0;
10 int cMl=0;
11 int cSl=0;
12 void resumen();
13 %}
14 numero   [0-9]
15 dominio  [a-z]+\.*
16 term     \.[a-z]{2,3}
17 correo   .+@{dominio}+{term}
18 pagina   "http://"?"www."?.+"."{dominio}*{term}("/".*)*
19 separador                ("/"|"-")
20 dia                              (0?[1-9]|[10-30]|31)
21 mes                              (0?[1-9]|[10-12])
22 anio                     ([0-9]{2}|[0-9]{4,})
23 fechaNormal              {dia}{separador}{mes}{separador}{anio}
24 fechaInglesa     {anio}{separador}{mes}{separador}{dia}
25 pre_int                  \+[0-9]{2,3}
26 movil                    (6|7)[0-9]{8}
27 fijo                     (8|9)[0-9]{8}
28 telefono                 {pre_int}+({fijo}|{movil})
29 matricula_t1     [0-9]{4}"-"?([B-D]|[F-H]|[J-N]|[P-T]|[V-Z]){3}
30 matricula_t2     [A-Z]{1,2}"-"?[0-9]{4}"-"?[A-Z]{2}
31 dni                              [X-Z]{1}-?[0-9]{7}-?[a-zA-Z]|[0-9]{8}-?[a-
       zA-Z]
32 cod_post                 (0[1-9]|[1-4][0-9]|5[0-2])[0-9]{3}
33 apertura_ml      "/\*"
34 cerradura_ml     "\*/"
35 comentario       [^"/\*"]
36 apertura_sl      "//"
37 ml_comment       {apertura_ml}{comentario}*{cerradura_ml}
38 sl_comment       {apertura_sl}.*
39
40 %%
41
42 {correo}          {printf("Encontrado un correo: %s\n", yytext);}
43 {pagina}          {printf("Encontrada una pagina: %s\n", yytext);}
44 {fechaNormal}     {cFechaNormal+=1;printf("Encontrada una fecha en formato dd
       /mm/aaaa: %s\n", yytext); REJECT;}
45 {fechaInglesa}    {cFechaIng+=1; printf("Encontrada una fecha en formato aaaa
       /mm/dd: %s\n", yytext);}
46 {telefono}                {cTlf+=1; printf("Encontrado un telefono: %s\n",
       yytext);}
47 {matricula_t1}   {cMat+=1; printf("Encontrada una matricula: %s\n", yytext)
       ;}
48 {matricula_t2}   {cMat+=1; printf("Encontrada una matricula: %s\n", yytext)
```

```
    ;}
49  {cod_post}              {cCP+=1; printf("Encontrado un codigo postal: %s\n"
    , yytext);}
50  {dni}                   {cDni+=1; printf("Encontrado un dni: %s\n", yytext)
    ;}
51  {ml_comment}    {cMl+=1;printf("Encontrado un comentario multilinea: %s\n",
     yytext);}
52  {sl_comment}    {cSl+=1;printf("Encontrado un comentario monolinea: %s\n",
    yytext);}
53  .                       {}
54
55  %%
56
57  int main(int argc, char* argv[]){
58      int archivos=0;
59      for(archivos=1;archivos<argc;archivos++){
60              if(argc>=2){
61                  yyin=fopen(argv[archivos],"rt");
62                  if(yyin==NULL){
63                          printf("El fichero %s no se puede abrir\n",argv[
                               archivos]);
64                          exit(-1);
65                  }
66              }else{yyin=stdin;}
67              yylex();
68          }
69      return 0;
70  }
71  void resumen(){
72          printf("Se han encontrado %d fechas en formato dd/mm/aa.\n",
               cFechaNormal);
73          printf("Se han encontrado %d fechas en formato aaaa/mm/dd.\n",
               cFechaIng);
74          printf("Se han encontrado %d numeros de telefono.\n", cTlf);
75          printf("Se han encontrado %d matriculas.\n", cMat);
76          printf("Se han encontrado %d codigos postales.\n", cCP);
77          printf("Se han encontrado %d dni.\n", cDni);
78
79          printf("Se han encontrado %d comentarios multilinea.\n", cMl);
80      printf("Se han encontrado %d comentarios monolinea.\n", cSl);
81  }
```