# Flex

## The Fast Lexical Analyzer

Μεταγλωττιστές
thp06@di.uoa.gr

Τμήμα Πληροφορικής & Τηλεπικοινωνιών
Ε.Κ.Π.Α

23 Μαρτίου 2009

# Φάση Λεκτικής Ανάλυσης (1/2)

- Είσοδος: μία συμβολοσειρά (πρόγραμμα)
- Έξοδος : λεκτικές μονάδες (tokens)

**Ορολογία**

*Token* : αναπαρίσταται από ένα ζεύγος (*tokenName*, *attribute*), όπου *attribute* ένας αριθμός/μία συμβολοσειρά/ένας δείκτης στον πίνακα συμβόλων/μία δομή και το οποίο είναι προαιρετικό. Ένα *tokenName* είναι ένα αφηρημένο σύμβολο (δηλ. δεν ανήκει στη γλώσσα) που εκφράζει μία κλάση χαρακτήρων (π.χ. **id**, **string**) και το οποίο αποτελεί την είσοδο του συντακτικού αναλυτή.

# Φάση Λεκτικής Ανάλυσης (2/2)

**Ορολογία**

*Lexeme* (ή λέξημα): μία ακολουθία χαρακτήρων που ταιριάζει με ένα *pattern* (π.χ. το *keyword if*, ο αριθμός 5 ή το *string* "hello" ).

*Pattern* (ή σχήμα): είναι μία περιγραφή της μορφής που μπορούν να πάρουν τα λεξήματα (π.χ. μία κανονική έκφραση).

**Λειτουργία λεκτικής ανάλυσης**: Διαβάζει *χαρακτήρες* και τους ταιριάζει με *σχήματα*, παράγοντας *λεξήματα*, τα οποία αντιστοιχίζονται (όχι 1-1) σε *λεκτικές μονάδες*, οι οποίες αποτελούν την είσοδο του συντακτικού αναλυτή.

# Παράδειγμα λεκτικής ανάλυσης

Είσοδος : ... if (cond) ...

Χαρακτήρες : ..., i, f, (, c, o, n, d, ), ...

Lexemes : ..., if, (, cond, ), ...

Tokens : ..., *keyword*, *L_PAR*, *id*, *R_PAR*, ...

Τί μορφή έχουν τα *patterns*;

# Κανονικές εκφράσεις (Θεωρία) (1/2)

- Χρησιμοποιούνται για να εκφράσουν σαφώς και περιεκτικά σύνολα γλωσσών.
- Προκύπτουν από την εφαρμογή των πράξεων της Ένωσης ($\cup$), Παράθεσης ($\cdot$) και **Kleene Star** ($*$) πάνω σε γλώσσες.
- Πολύ χρήσιμες στους λεκτικούς αναλυτές για την αναγνώριση ακολουθιών χαρακτήρων ως κλάσεις (π.χ. αναγνωριστικά, ακέραιοι, σταθερές συμβολοσειρές, κ.α.).

Τα *patterns* προσδιορίζονται με Κ.Ε.

# Κανονικές εκφράσεις (Θεωρία) (2/2)

Έστω $\Sigma$ ένα αλφάβητο (το $\Sigma$ μπορεί να θεωρηθεί και ως γλώσσα).

- Αν $a \in \Sigma$, τότε $\mathbf{a}$ είναι μία Κ.Ε. που αναπαριστά τη γλώσσα $\{a\}$.
- Η κενή συμβολοσειρά $\epsilon$ είναι μία Κ.Ε. που αναπαριστά τη γλώσσα $\{\epsilon\}$.
- Αν $\mathbf{l}, \mathbf{r}$ Κ.Ε. που παριστάνουν τις γλώσσες $L$, $R$, τότε $\mathbf{l} \cup \mathbf{r}$, $\mathbf{l} \cdot \mathbf{r}$ και $\mathbf{l}^*$ είναι Κ.Ε. που παριστάνουν τις γλώσσες $L \cup R$, $L \cdot R$ και $L^*$ αντίστοιχα.

# Flex (1/2)

- Ο *Flex* είναι μία **γεννήτρια** λεκτικών αναλυτών (ή *scanners*) για $C/C++$.
- Υλοποίηση του *Lex* του συστήματος *Unix*.
- *Flex* = **F***ast* **Lex***ical Analyzer Generator*.

# Flex (2/2)

Είσοδος : ένα αρχείο στο οποίο δίνονται *κανόνες*, ώστε να αναγνωρισθούν τα *λεξήματα* να παραχθούν οι αντίστοιχες *λεκτικές μονάδες* και να γίνουν οι επιθυμητές ενέργειες.

Έξοδος : ένα αρχείο με όνομα *lex.yy.c* στο οποίο υπάρχει ο κώδικας που υλοποιεί το λεκτικό αναλυτή.

# Κανονικές Εκφράσεις στον Flex (1/3)

Ο *Flex* υποστηρίζει μία επέκταση των Κ.Ε.:

- $x$ : ο χαρακτήρας $x$.
- . : οποιοσδήποτε χαρακτήρας εκτός του χαρακτήρα νέας γραμμής.
- $"abc"$ : η ακολουθία χαρακτήρων $abc$.
- $[abc]$ : οποιονδήποτε από τους χαρακτήρες $a, b, c$.
- $[a-z]$ : οποιονδήποτε από τους χαρακτήρες $a$ έως $z$.
- $[^a-z]$ : οποιονδήποτε χαρακτήρα εκτός από τους χαρακτήρες $a$ έως $z$.
- $r^*$ : καμία ή περισσότερες επαναλήψεις της Κ.Ε. $r$.
- $r^+$ : μία ή περισσότερες επαναλήψεις της Κ.Ε. $r$.
- $r?$ : μία ή καμία επανάληψη της Κ.Ε. $r$.

# Κανονικές Εκφράσεις στον Flex (2/3)

$r\{i,j\}$ : από $i$ εως $j$ $(0 < i < j)$ επαναλήψεις της Κ.Ε. $r$.

$rs$ : ακολουθίες που προκύπτουν από την παράθεση των Κ.Ε. $r$ και $s$.

$(r)$ : οι παρενθέσεις ορίζουν την εφαρμογή των τελεστών.

$r|s$ : ακολουθίες που ικανοποιούν ή την $r$ ή την $s$.

$^r$ : ικανοποιείται μόνο αν η Κ.Ε. $r$ βρίσκεται στην αρχή της γραμμής.

$r$\$ : ικανοποιείται μόνο αν η Κ.Ε. $r$ βρίσκεται στο τέλος της γραμμής.

$<< EOF >>$ : ικανοποιείται όταν συναντηθεί $EOF$.

Για τη χρήση των παραπάνω ειδικών χαρακτήρων \$, $\wedge$, |, ), ( κ.τ.λ. ως κυριολεκτικών θα πρέπει να προηγηθεί ο χαρακτήρας $'\backslash'$.

# Κανονικές Εκφράσεις στον Flex (3/3)

Τέλος, υποστηρίζονται αρκετές κλάσεις χαρακτήρων, όπως οι:

- [ : *lower* : ] : οποιοσδήποτε πεζός λατινικός χαρακτήρας.
- [ : *upper* : ] : οποιοσδήποτε κεφαλαίος λατινικός χαρακτήρας.
- [ : *alpha* : ] : οποιοσδήποτε πεζός ή κεφαλαίος λατινικός χαρακτήρας.
- [ : *digit* : ] : οποιοδήποτε δεκαδικό ψηφίο.
- [ : *blank* : ] : ο κενός χαρακτήρας ή ο στηλοθέτης.
- [ : *print* : ] : οποιοσδήποτε εκτυπώσιμος χαρακτήρας συμπεριλαμβανομένου του κενού.
- [ : *xdigit* : ] : οποιοδήποτε δεκαεξαδικό ψηφίο.

# Δομή προγράμματος Flex (1/5)

Ένα αρχείο *Flex* αποτελείται από 3 μέρη που διαχωρίζονται από '%%'.

```
Ορισμοί
%%
Κανόνες
%%
Κώδικας Χρήστη
```

# Δομή προγράμματος Flex (2/5)

Οι **ορισμοί** αποτελούνται από:

- ορισμούς ονομάτων που χρησιμοποιούνται ως συντομογραφίες Κ.Ε., και έχουν την μορφή:

```
name    regular_expression
```

π.χ.

```
digit   [0-9]
id      [a-zA-Z][a-zA-Z_0-9]*
```

# Δομή προγράμματος Flex (3/5)

- κώδικα που θέλουμε να συμπεριληφθεί στον παραγόμενο λεκτικό αναλυτή, συνήθως δηλώσεις μακροεντολών, μεταβλητών και τύπων δεδομένων, ανάμεσα σε ′%{ %}′ π.χ.

```
%{
    #include <stdio.h>

    #define TK_IF   0
    #define TK_ELSE 1

    typedef struct {
        char lexeme[256];
        int lineNumber, charPosition;
    } tokenInfo;
%}
```

# Δομή προγράμματος Flex (4/5)

Το τμήμα **κανόνων** είναι το κύριο τμήμα του προγράμματος και αποτελείται από κανόνες της μορφής:

```
pattern   {action}
```

π.χ.

```
%%
{id}          {printf("Found id: %s\n",yytext)
{digit}       {printf("Found digit: %s\n", yytext);
%%
```

* Όταν ικανοποιείται ένα *pattern*, ο κανόνας ενεργοποιείται και ο κώδικας που ακολουθεί εκτελείται.

# Δομή προγράμματος Flex (5/5)

* Όταν ικανοποιούνται περισσότεροι από ένας κανόνες, τότε επιλέγεται αυτός που καταναλώνει περισσότερους χαρακτήρες.
* Εάν καταναλώνουν τον ίδιο αριθμό, επιλέγεται αυτός που έχει δηλωθεί πρώτος.

Το τμήμα **κώδικα χρήστη** είναι προαιρετικό και ο σκοπός του είναι η άμεση υλοποίηση συναρτήσεων που χρησιμοποιούνται στον παραγόμενο λεκτικό αναλυτή (ο κώδικας αντιγράφεται χωρίς αλλαγές).

# Περιβάλλον Flex

Μερικές από τις κυριότερες διαθέσιμες μεταβλητές/συναρτήσεις:

*char * yytext* : περιέχει το κομμάτι του κειμένου που έχει ικανοποιήσει την κανονική έκφραση (*lexeme*).

*int yyleng* : ένας ακέραιος που δηλώνει το μέγεθος του *yytext*.

*FILE * yyin* : το προκαθορισμένο αρχείο εισόδου (μπορούμε να το αλλάξουμε).

*int yylex()* : η παραγόμενη συνάρτηση λεκτικής ανάλυσης· επιστρέφει το αναγνωριστικό της λεκτικής μονάδας, που διαβάζει.

# Σχόλια στον Flex

Ο *Flex* υποστηρίζει σχόλια της μορφής /* ... */, τα οποία αντιγράφονται αυτούσια στο παραγόμενο πρόγραμμα *C*. Μπορούν να βρίσκονται οπουδήποτε εκτός από:

* τις γραμμές που ξεκινούν με %*option* και
* την αρχή των γραμμών του τμήματος των κανόνων.

# Καταστάσεις στον Flex (1/2)

Ο *Flex* υποστηρίζει υπό συνθήκη ενεργοποίηση ενός κανόνα μέσω της έννοιας της **κατάστασης**. Αρχικά ο *Flex* βρίσκεται στην κατάσταση *INITIAL*.
Οι καταστάσεις δηλώνονται στο τμήμα ορισμών. (π.χ. *%s A_STATE*).

Ένας υπό συνθήκη ενεργός κανόνας γράφεται ως εξής:

```
<A_STATE>regular_expression    action
```

# Καταστάσεις στον Flex (2/2)

ή ως:

```
<A_STATE,ANOTHER_STATE>regular_expression    action
```

,όπου ο κανόνας είναι ενεργός όντας σε οποιαδήποτε από τις δύο καταστάσεις.

Η μετάβαση σε μία κατάσταση γίνεται γράφοντας στο τμήμα της ενέργειας :

```
BEGIN(A_STATE);
```

Πολύ χρήσιμη δυνατότητα για σχόλια πολλών γραμμών.

# Παράδειγμα προγράμματος Flex (1/3)

```
%{
    /* definitions of manifest constants LT, LE,
    EQ,NE, GT, GE, IF, ELSE, ID, NUMBER, RELOP */
    #define TK_LT 257
    #define TK_LE 258
    ...
%}

/* regular definitions */
delim     [ \t\n]
ws        {delim}+
letter    [A-Za-z]
digit     [0-9]
id        {letter}({letter}|{digit})*
number    {digit}+(\.{digit}+)?(E[+-]?{digit}+)?
```

# Παράδειγμα προγράμματος Flex (2/3)

```
%%

{ws}      {/*no action and no return */}
if        {return(TK_IF);}
else      {return(TK_ELSE);}
{id}      {yylval = (int) installID(); return(TK_ID);}
{number}  {yylval = (int) installNum(); return(TK_NUMBER);}
"<"       {yylval = TK_LT; return(RELOP);}
">"       {yylval = TK_GT; return(RELOP);}

%%

int installID() {
/* function to install the lexeme into the
 * symbol table and return a pointer */
}
```

# Παράδειγμα προγράμματος Flex (3/3)

```
int installNum() {
/* similar to installID, but puts numerical
 * constants to a separate table */
}

int main() {

    int token;
    while ((token = yylex()) != TK_EOF)
        printf("Token %d: %s\n", token, yytext);

    return 0;
}
```

# Διάγνωση και Χειρισμός Σφαλμάτων

**Διάγνωση**

- Χαρακτήρες που δεν ανήκουν στη γλώσσα (π.χ. @).
- Χαρακτήρες που δεν είναι έγκυροι ως προς ένα *token* (π.χ. var#cont ).

**Χειρισμός**

- Αναφορά σφάλματος και συνέχεια λεκτικής ανάλυσης από την αμέσως επόμενη ακολουθία χαρακτήρων που αντιστοιχεί σε μία λεκτική μονάδα.
- *Panic – mode*: αναφορά σφάλματος και άμεσος τερματισμός.
- Εισαγωγή ειδικών error-token και συγγραφή ξεχωριστών κανόνων σφάλματος από το συντακτικό αναλυτή.
- Διόρθωση σφάλματος (σπάνια υλοποιείται).

# Διαδικασία παραγωγής λεκτικού αναλυτή

Το αρχείο περιγραφής (*floop2009.l*) διοχετεύεται ως είσοδος στη γεννήτρια κώδικα λεκτικής ανάλυσης.

$$\$flex \;\; floop2009.l$$

Ο κώδικας που προκύπτει ως αποτέλεσμα περιλαμβάνεται στο αρχείο *lex.yy.c* και περιλαμβάνει τη συνάρτηση λεκτικής ανάλυσης *yylex*().
Αν το αρχείο περιγραφής περιλαμβάνει συνάρτηση *main*(), τότε το πρόγραμμα που παράχθηκε μπορεί να λειτουργήσει αυτόνομα και για να γίνει αυτό πρέπει να περάσει από ένα μεταγλωττιστή της *C*.

$$\$gcc \;\; lex.yy.c \;\; -lfl$$

# Αναφορές (1/2)



Νικόλαος Σ. Παπασπύρου, Εμμανουήλ Σ. Σκορδαλάκης.
Μεταγλωττιστές
*Εκδόσεις Συμμετρία. 2002.*


Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ullman.
Compilers: Principles, Techniques, and Tools.
*Addison-Wesley. Second Edition, 2007.*

# Αναφορές (2/2



Charles Fischer, University of Wisconsin, Richard LeBlanc.
Crafting a Compiler with C.
*Addison-Wesley, 1991.*


Εγχειρίδιο χρήσης *Flex*.
`http://flex.sourceforge.net/manual/` .