ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επεξεργασία Βιβλιογραφικών Βάσεων BibTeX

Μαυρουδής Δημήτριος

Επιβλέπων Καθηγητής: Βασίλειος Δημακόπουλος

Ιωάννινα, Μάιος 2007

Περίληψη στα ελληνικά

Στο σύγγραμα αυτό περιγράφουμε το σχεδιασμό και την υλοποίηση μιας βιβλιοθήκης, η οποία παρέχει πρόσβαση και διαχείριση σε βιβλιογραφικές βάσεις που ακολουθούν το πρότυπο του BibTeX. Η βιβλιοθήκη αυτή αναλύει λεκτικά και συντακτικά τα BibTeX αρχεία, αναγνωρίζει τις καταχωρήσεις αυτών, και παρέχει ένα API (Application Programming Interface), δηλαδή ένα σύνολο συναρτήσεων γραμμένες σε γλώσσα προγραμματισμού C, για την επεξεργασία αυτών. Έτσι πληρεί τις προδιαγραφές του BibTeX στο σύνολό του.

Πίνακας περιεχομένων

1

Εισαγωγή

Στο κεφάλαιο αυτό γίνεται μια εισαγωγή στο θέμα αυτής της πτυχιακής εργασίας. Αρχικά περιγράφεται το σύστημα στοιχειοθέτησης εγγράφων TeX. Αναφέρεται το που, από ποιούς και γιατί χρησιμοποιείται. Στη συνέχεια περιγράφεται με τον ίδιο τρόπο και το LaTeX το οποίο στηρίζεται πάνω στο TeX. Ακολουθεί η γενική περιγραφή του BibTeX και στη συνέχεια μια περίληψη του τρόπου λειτουργίας του. Έτσι φτάνουμε και στο σκοπό της πτυχιακής εργασίας που σχετίζεται άμεσα με τα αρχεία τύπου BibTeX (*.bib). Τέλος, στη 2^{η} ενότητα γίνεται ένας πρόλογος για τη γλώσσα προγραμματισμού C και στην 3^{η} ενότητα του κεφαλαίου περιγράφεται η οργάνωση των κεφαλαίων αυτού του συγγράματος.

1.1 Περιγραφή του Θέματος

Η ενότητα αυτή πραγματεύεται σε περίληψη το αντικείμενο της πτυχιακής (ποιος ο στόχος και οι ανάγκες καθώς και ίσως κάποιες απαραίτητες έννοιες που είναι απαραίτητες για την κατανόηση των παρακάτω). Πριν προχωρήσουμε όμως, καλό θα ήταν να κάνουμε μια μικρή εισαγωγή στο TeX και στο LaTeX και στη συνέχεια και στο BibTeX πριν αναφερθούμε στο αντικείμενο της πτυχιακής.

1.1.1 To TeX[4, 22-30]

Το TeX είναι ένα πρόγραμμα στοιχειοθεσίας εγγράφων που αναπτύχθηκε από τον Donald Knuth. Μαζί με τη γλώσσα για την περιγραφή των γραμματοσειρών και τους μοντέρνους εκτυπώσιμους χαρακτήρες των υπολογιστών, σχεδιάστηκε για 2 κυρίως σκοπούς:

- Να επιτρέπει σε κάθε χρήστη να παράγει υψηλής ποιότητας βιβλία καταβάλλοντας μια μέτρια μόλις προσπάθεια.
- Να παρέχει ένα σύστημα τέτοιο που να δίνει το ίδιο ακριβώς αποτέλεσμα σε όλους τους υπολογιστές, τόσο τώρα όσο και στο μέλλον.

Το TeX είναι ελεύθερο λογισμικό (δωρεάν) και πολύ δημοφιλές στους ακαδημαϊκούς κύκλους, και πιο ειδικά στην κοινωνία των μαθηματικών, της φυσικής, της επιστήμης των υπολογιστών, των οικονομικών, των πολιτικών επιστημών και των μηχανικών. Θεωρείται από μερικούς ως ο καλύτερος δυνατός τρόπος για τη στοιχειοθέτηση περίπλοκων μαθηματικών τύπων, αλλά πλέον χρησιμοποιείται και για άλλους στοιχειοθετικούς σκοπούς, ειδκά με τη μορφή του LaTeX. Παίρνει ένα κατάλληλα προετοιμασμένο αρχείο υπολογιστή και το μετατρέπει σε έναν τύπο αρχείου το οποίο μπορεί να εκτυπωθεί από πολλά είδη εκτυπωτών.

1.1.2 To LaTeX[2, 18-21]

Το LaTeX είναι ουσιαστικά μία από τις «διαλέκτους» του TeX. Χρησιμοποιείται ευρέως από μαθηματικούς, επιστήμονες, φιλοσόφους, μηχανικούς και γενικά πανεπιστημιακούς, καθώς και από τον εμπορικό κόσμο, ως ένα πρωτεύον ή ένα ενδιάμεσο σύστημα τυποποίησης λόγω της ποιότητας στη στοιχειοθέτηση που επιτυγχάνεται από το TeX. Πρόκειται για ένα σύστημα προετοιμασίας εγγράφων με υψηλή ποιότητα στοιχειοθέτησης, ιδιαίτερα βολικό για την παραγωγή μεγάλων άρθρων και βιβλίων. Σκοπός του να παρέχει μια υψηλού επιπέδου γλώσσα πρόσβασης στη δύναμη του TeX. Έτσι, επειδή οι εντολές στοιχειοθέτησης του TeX είναι πολύ χαμηλού επιπέλου είναι συνήθως πιο απλό για τον τελικό χρήστη να χρησιμοποιήσει το LaTeX. Το LaTeX γράφτηκε στις αρχές της δεκαετίας του 80 από τον Leslie Lamport και έχει γίνει η πλέον διαδεδομένη μέθοδος μεταχείρισης του TeX. Όπως και το TeX έτσι και το LaTeX είναι ελεύθερο λογισμικό (δωρεάν).

1.1.3 Το BibTeX[1, 3]

Πρόκειται για ένα εργαλείο στοιχειοθέτησης μιας λίστας αναφορών-παραπομπών. Τυπικά χρησιμοποιείται σε συνδυασμό με το LaTeX κάνοντας εύκολη την παράθεση πηγών με συνέπεια κι αξιοπιστία, διαχωρίζοντας τις βιβλιογραφικές πληροφορίες από την εμφάνιση αυτών. Στην ίδια αρχή διαχωρισμού του περιεχομένου και της εμφάνισης αυτού στηρίζεται και το ίδιο το LaTeX. Το BibTeX δημιουργήθηκε από τον Oren Patashnik και τον Leslie Lamport το 1985.

1.1.4 Σκοπός της Πτυχιακής

Το BibTeX χρησιμοποιεί έναν τύπο αρχείου βασισμένο σε απλό κείμενο ανεξαρτήτου στυλ για τη λίστα των βιβλιογραφικών αντικειμένων, όπως άρθρα, βιβλία, κ.λ.π. Διαβάζει το βοηθητικό αρχείο *.aux, που βρίσκεται στο επίπεδο κορυφής και δημιουργείται κατά το πρώτο τρέξιμο του TeX/LaTeX, και δημιουργεί ένα αρχείο βιβλιογραφίας (αρχείο *.bbl). Η λειτουργία του έχει ως εξής: Αναζητά στα αρχεία βιβλιογραφικών βάσεων (αρχεία *.bib) που ορίζονται από την εντολή \bibliography του LaTeX τις καταχωρήσεις-εγγραφές που καθορίζονται απο τις εντολές \cite και \nocite του πηγαίου αρχείου LaTeX ή TeX. Τις πληροφορίες που παίρνει από αυτές τις καταχωρήσεις-εγγραφές τις τυποποιεί με βάση τις οδηγίες ενός αρχείου βιβλιογραφικού στυλ (αρχείο *.bst), το οποίο καθορίζεται από την εντολή \bibliographystyle του LaTeX, και παράγει τα αποτελέσματα στο αρχείο βιβλιογραφίας (αρχείο *.bbl). Το BibTeX δουλεύει σε 2 κύρια στάδια:

- Αναγνώριση (εύρεση) των εγγραφών από τη βιβλιογραφική βάση δεδομένων.
- Εμφάνιση αυτών με κάποιο συγκεκριμένο τρόπο (στυλ).

Το πρώτο στάδιο είναι αυτό που θα μας απασχολήσει στη συνέχεια και αποτελεί το στόχο αυτής της πτυχιακής εργασίας.

Πιο συγκεκριμένα λοιπόν, σκοπός μας είναι η δημιουργία ενός API (Application Programming Interface) που θα παρέχει στο χρήστη-προγραμματιστή κάποιες βασικές λειτουργίες για την αναγνώριση των εγγραφών μιας βιβλιογραφικής βάσης BibTeX. Αυτό θα γίνει μέσα από κάποιες συναρτήσεις ενσωματωμένες σε μια στατική βιβλιοθήκη που θα φτιάξουμε και θα είναι γραμμένες σε γλώσσα προγραμματισμού C. Έτσι η λειτουργία των συναρτήσεων αυτού του API θα γίνεται σε 4 βασικά στάδια:

I. Συντακτική ανάλυση της βιβλιογραφικής βάσης BibTeX.
II. Αναγνώριση όλων των εγγραφών του BibTeX.
III. Εύρεση (εξαγωγή) συγκεκριμένων εγγραφών με βάση κάποιο κλειδί.

IV. Αποθήκευση των εγγραφών αυτών σε μια κατάλληλη δομή δεδομένων για περαιτέρω επεξεργασία από τον χρήστη-προγραμματιστή.

1.2 Η Γλώσσα Προγραμματισμού C

Όπως προαναφέρθηκε, η δημιουργία του API θα γινει σε γλώσσα προγραμματισμού C. Η C είναι μιας γενικής χρήσης γλώσσας προγραμματισμού που χαρακτηρίζεται από οικονομία στην έκφραση, σύγχρονη ροή ελέγχου και δομές δεδομένων, κι ένα πλούσιο σύνολο τελεστών. Η C δεν είναι γλώσσα «πολύ υψηλού επιπέδου», ούτε και «εκτεταμένη», και δεν εξειδικεύεται σε κανένα συγκεκριμένο πεδίο εφαρμογών. Όμως, η απουσία περιορισμών και η γενικότητά της, την καθιστούν πιο άνετη και πιο αποτελεσματική για πολλές εργασίες, από υποτιθέμενες ισχυρότερες γλώσσες προγραμματισμού. Έτσι λοιπόν, δε θα μπορούσαμε κι εμείς παρά να χρησιμοποιήσουμε τη C για την υλοποίηση της πτυχιακής εργασίας.

1.3 Οργάνωση του Συγγράματος

Στην ενότητα αυτή περιγράφεται η οργάνωση του συγγράματος αυτού, δηλαδή το περιεχόμενο του κάθε κεφαλαίου. Στο 1^{ο} κεφάλαιο γίνεται μια μικρή αναφορά στο TeX, το LaTeX και το BibTeX. Στη συνέχεια περιγράφεται ο στόχος της πτυχιακής εργασίας και κλείνει με μια αναφορά στη γλώσσα προγραμματισμού C. Στο 2^{ο} κεφάλαιο γίνεται μια πιο εκτενής περιγραφή του BibTeX, όπου περιγράφονται αναλυτικά τα πεδία και οι καταχωρήσεις του, μαζί με κάποια χαρακτηριστικά του και συνοδευτικά παραδείγματα. Έπειτα αναφέρονται τα πιο σημαντικά εργαλεία που ασχολούνται με την επεξεργασία βιβλιογραφικών βάσεων BibTeX, οπότε και γίνεται μια γενική σύγκριση με την εργασία αυτή. Στο 3^{ο} κεφάλαιο ακολουθεί ο σχεδιασμός του συντακτικού αναλυτή που υλοποιήθηκε. Πιο συγκεκριμένα περιγράφεται η γλώσσα που αναγνωρίζεται από τον λεκτικό αναλυτή, καθώς και η γραμματική που αναγνωρίζει ο συντακτικός αναλυτής. Έπειτα δίνονται οι δομές δεδομένων που χρησιμοποιούνται για την αποθήκευση των καταχωρήσεων της βιβλιογραφικής βάσης BibTeX. Στο 4^{ο} κεφάλαιο πλέον φτάνουμε και στη βασική υλοποίηση της βιβλιοθήκης. Εκεί δίνονται οι υλοποιημένες συναρτήσεις μαζί με αναλυτική περιγραφή για το τι κάνει η κάθε μια. Το κεφάλαιο τελειώνει με τα προγραμματιστικά εργαλεία που απαιτούνται για να δημιουργηθούν τα εκτελέσιμα αρχεία που περιλαμβάνει η βιβλιοθήκη. Τέλος στο 5^{ο} κεφάλαιο περιγράφεται ο τρόπος χρήσης της βιβλιοθήκης, πως μπορεί δηλαδή

να ενσωματωθεί στον κώδικα του χρήστη-προγραμματιστή, μέσα από 2 απλά παραδείγματα σε C, και τελειώνει με μια σύνοψη όπου λέμε απλά ότι φτιάξαμε τη βιβλιοθήκη που θέλαμε. Στο 6^{ο} κεφάλαιο δίνεται η βιβλιογραφία και στο παράρτημα Α ο πηγαίος κώδικας της βιβλιοθήκης.

2

BibTeX και Σχετικά Εργαλεία

Σε αυτό το κεφάλαιο περιγράφονται πιο λεπτομερώς τα στοιχεία μιας βιβλιογραφικής βάσης BibTeX. Περιγράφονται οι κατηγορίες των καταχωρήσεων σε μια βιβλιογραφική βάση, καθώς και οι τύποι των πεδίων αυτών. Επίσης αναφέρονται κάποια ιδιόρυθμα χαρακτηριστικά ενός αρχείου BibTeX.Τέλος αναφέρονται κάποια, σχετικά με τη πτυχιακή αυτή εργασία, εργαλεία με τη βασική τους λειτουργία και γίνεται μια μικρή και γενική σύγκριση.

2.1 Η Βιβλιογραφική Βάση BibTeX

Σ'αυτήν την ενότητα θα δούμε λίγο πιο αναλυτικά τα περιεχόμενα ενός αρχείου BibTeX. Θα δούμε τη γενική σύνταξή του, τα διάφορα χαρακτηριστικά που υποστηρίζει και τις λειτουργίες που εκτελεί. Επίσης θα δούμε τα είδη των καταχωρήσεων που αναγνωρίζονται από το BibTeX, καθώς και τα είδη των πεδίων για τις διάφορες καταχωρήσεις. Τέλος θα δείξουμε κάποια παραδείγματα αρχείων BibTeX για την καλύτερη κατανόηση όλων των παραπάνω.

2.1.1 Γενική Δομή

Το BibTeX χρησιμοποιεί έναν τύπο αρχείου βασισμένο σε απλό κείμενο ανεξαρτήτου στυλ για τη λίστα των βιβλιογραφικών αντικειμένων, όπως άρθρα, βιβλία, κ.λ.π. Μπορεί να περιέχει μία ή και περισσότερες καταχωρήσεις σε ένα αρχείο. Κάθε καταχώρηση ορίζεται με τον ειδικό χαρακτήρα '@' ακολουθούμενο από τον τύπο της καταχώρησης. Στη συνέχεια ακολουθεί σε μορφή δομής, δηλαδή μέσα σε { } ή ( ), το κλειδί της καταχώρησης και η λίστα με τα πεδία αυτής. Η γενική μορφή δηλαδή μιας καταχώρησης είναι:

```
@<τύπος καταχώρησης>{<κλειδί>,
<πεδίο 1> = <τιμή>,
<πεδίο 2> = <τιμή>,
...
<πεδίο n> = <τιμή>
}
```

Επίσης με τη βοήθεια της ειδικής καταχώρησης `@STRING` μπορούν να ορισθούν διάφορες βοηθητικές μακροεντολές για κάθε αρχείο. Π.χ:

```
@STRING{X="Jack Black"}
@ARTICLE{A1,
author = X,
…}
@BOOK{B1,
editor = X,
…}
```

2.1.2 Τα Πεδία

Αναφορές σε διαφορετικά είδη δημοσιευμάτων περιέχουν και διαφορετικές πληροφορίες. Έτσι καταχωρήσεις διαφορετικών κατηγοριών έχουν και διαφορετικά πεδία. Για το λόγο αυτό δε θα μπορούσε και το BibTeX παρά να υποστηρίζει μια πληθώρα τέτοιων πεδίων για την καλύτερη περιγραφή των καταχωρήσεων. Για κάθε κατηγορία καταχώρησης τα πεδία χωρίζονται σε 3 κλάσεις:

1) Απαιτούμενα (Required). Η παράληψη ενός τέτοιου πεδίου θα έχει σαν αποτέλεσμα κάποια προειδοποίηση και, σπάνια, μια κακώς στοιχειοθετημένη βιβλιογραφική καταχώρηση.

2) Προαιρετικά (Optional). Αν υπάρχει η πληροφορία του πεδίου θα χρησιμοποιηθεί, ενώ μπορεί κάλλιστα να παραληφθεί χωρίς να προκαλέσει στοιχειοθετικά προβλήματα.
3) Αδιάφορα (Ignored). Το πεδίο αυτό αγνοείται. Το BibTex αγνοεί κάθε πεδίο που δεν είναι ούτε απαιτούμενο ούτε προαιρετικό.

Παρακάτω δίνεται μια περιγραφή όλων των διαθέσιμων πεδίων:

- **αddress**. Συνήθως η διεύθυνση του εκδότη ή κάποιου άλλου τύπου ίδρυμα.
- **annote**. Σχολιασμός.
- **author**. Το όνομα (ή ονόματα) του συγγραφέα (των συγγραφέων αντίστοιχα).
- **booktitle**. Ο τίτλος ενός βιβλίου, μέρος του οποίου αναφέρεται.
- **chapter**. Ο αριθμός ενός κεφαλαίου (ή ενότητας ή οτιδήποτε).
- **crossref**. Το κλειδί της καταχώρησης με την οποία διασταυρώνεται.
- **edition**. Η έκδοση ενός βιβλίου (αριθμητικό).
- **editor**. Ο εκδότης ή οι εκδότες.
- **howpublished**. Πώς εκδόθηκε κάτι αξιοπερίεργο.
- **institution**. Το ίδρυμα-χορηγός μιας τεχνικλης αναφοράς.
- **journal**. Μια ημερησία εφημερίδα.
- **key**. Χρησιμοποιείται για διασταυρώσεις καταχωρήσεων ή ως κριτήριο για να μπαίνουν σε αλφαβητική σειρά.
- **month**. Ο μήνας που δημοσιεύτηκε μια εργασία ή που γράφτηκε.
- **note**. Σημείωση, επιπρόσθετη πληροφορία.
- **number**. Ο αριθμός μιας ημερίδας, περιοδικού, τεχνικής αναφοράς ή μιας εργασίας σε σειρά εκδοτική.
- **organization**. Ο οργανισμός που χορηγεί μια σύσκεψη ή που δημοσιεύει ένα εγχειρίδιο.
- **pages**. Έναν ή και περισσότερους αριθμούς σελίδων ή γενικά μιας σειράς σελίδων.
- **publisher**. Το όνομα του εκδότη.
- **school**. Το όνομα ενός σχολείου όπου γράφτηκε μια διατριβή.
- **series**. Το όνομα μιαςς σειράς ή ένα σετ βιβλίων.
- **title**. Ο τίτλος της εργασίας.
- **volume**. Ο τόμος μιας ημερίδας ή ενός πολύτομου βιβλίου.

- **year**. Το έτος δημοσίευσης ή εγγραφής μιας εργασίας.

2.1.3 Οι Καταχωρήσεις

Όπως προαναφέραμε, το BibTeX υποστηρίζει ένα μεγάλο πλήθος από είδη καταχωρήσεων. Κατά την εισαγωγή μιας καταχώρησης λοιπόν στη βιβλιογραφική βάση BibTeX, το πρώτο πράγμα που πρέπει να αποφασιστεί είναι ο τύπος αυτής της καταχώρησης. Φυσικά κανένα σχήμα ταξινόμησης δεν είναι πλήρες, αλλά το BibTeX παρέχει αρκετούς τύπους καταχωρήσεων για να χειριστεί πολύ καλά σχεδόν κάθε αναφορά-παραπομπή. Ας δούμε τώρα τις κατηγορίες των καταχωρήσεων μαζί με τα απαιτούμενα και προαιρετικά πεδία της κάθε μιας:

- **`@Article`**. Ένα άρθρο μιας εφημερίδας ή ενός περιοδικού. Απαιτούμενα πεδία: author, title, journal, year. Προαιρετικά πεδία: volume, number, pages, month, note.
- **`@Book`**. Ένα βιβλίο με έναν σαφή εκδότη. Απαιτούμενα πεδία: author ή editor, title, publisher, year. Προαιρετικά πεδία: volume ή number, series, address, edition, month, note.
- **`@Booklet`**. Μια εργασία που εκτυπώθηκε και δεσμεύτηκε, χωρίς προκαθορισμένο εκδότη ή χορηγό. Απαιτούμενα πεδία: title. Προαιρετικά πεδία: author, howpublished, address, month, year, note.
- **`@Conference`**. Όπως και το INPROCEEDINGS.
- **`@Inbook`**. Το μέρος ενός βιβλίου, το οποίο μπορεί να είναι ένα κεφάλαιο, μια ενότητα ή/και μια σειρά σελίδων. Απαιτούμενα πεδία: author ή editor, title, chapter και/ή pages, publisher, year. Προαιρετικά πεδία: volume ή number, series, type, address, edition, month, note.
- **`@Incollection`**. Το μέρος ενός βιβλίου με δικό του τίτλο. Απαιτούμενα πεδία: author, title, booktitle, publisher, year. Προαιρετικά πεδία: editor, volume ή number, series, type, chapter, pages, address, edition, month, note.
- **`@Inproceedings`**. Ένα άρθρο στα πρακτικά μιας συνεδρίασης. Απαιτούμενα πεδία: author, title, booktitle, year. Προαιρετικά πεδία: editor, volume ή number, series, pages, address, month, organization, publisher, note.
- **`@Manual`**. Τεχνική τεκμηρίωση. Απαιτούμενα πεδία: title. Προαιρετικά πεδία: author, organization, address, edition, month, year, note.
- **`@Masterthesis.`** Μεταπτυχιακή διατριβή. Απαιτούμενα πεδία: author, title, scholl, year. Προαιρετικά πεδία: type, address, month, note.

- **@Misc**. Αν δεν είναι κάτι από όλα τα άλλα. Απαιτούμενα πεδία: κανένα. Προαιρετικά πεδία: author, title, howpublished, month, year, note.
- **@Phdthesis**. Διδακτορική διατριβή. Απαιτούμενα πεδία: author, title, school, year. Προαιρετικά πεδία: type, address, month, note.
- **@Proceedings**. Τα πρακτικά μιας συνεδρίασης. Απαιτούμενα πεδία: title, year. Προαιρετικά πεδία: editor, volume ή number, series, address, month, organization, publisher, note.
- **@Techreport**. Μια αναφορά που δημοσιεύτηκε από ένα σχολείο ή κάποιο άλλο ίδρυμα. Απαιτούμενα πεδία: author, title, institution, year. Προαιρετικά πεδία: type, number, address, month, note.
- **@Unpublished**. Μια τεκμηρίωση με συγγραφέα και τίτλο που δεν έχει δημοσιευτεί επίσημα. Απαιτούμενα πεδία: author, title, note. Προαιρετικά πεδία: month, year.

2.1.4 Χαρακτηριστικά – Λειτουργίες

Στην ενότητα αυτή θα αναφέρουμε κάποια βασικά χαρακτηριστικά και λειτουργίες του BibTeX. Θα αναφερθούμε μόνο σε αυτά βεβαίως που μας ενδιαφέρουν άμεσα, αφού θα πρέπει να συμπεριληφθούν και να ενσωματωθούν στην εργασία μας, και θα παραλείψουμε εκείνα που έχουν να κάνουν κυρίως με τα styles του BibTeX και δεν αποτελούν αντικείμενο της πτυχιακής μας εργασίας.

Πρώτο και πιο σημαντικό χαρακτηριστικό του BibTeX είναι η αλληλουχία (concatenation) των αλφαριθμητικών. Έτσι στην τιμή ενός πεδίου μπορεί να υπάρχει η αλληλουχία πολλών αλφαριθμητικών. Το BibTeX υποστηρίζει τον ορισμό μακροεντολών με τη βοήθεια του «ειδικού τύπου καταχώρησης» @STRING, οπότε μπορούμε να έχουμε το εξής:

```
@STRING(WGA = "Wolrd Gnus Almanac")
```

κι επομένως είναι εύκολο να παράγουμε σχεδόν ίδιους τίτλους πεδίων για διαφορετικές καταχωρήσεις. Για παράδειγμα:

```
@BOOK(almanac-66,
title = 1966 # WGA,
…)
@BOOK(almanac-67,
title = 1967 # WGA,
```

…)

κ.ο.κ. Ή θα μπορούσαμε να έχουμε:

```
month = "1~" # jan,
```

το οποίο και θα έδινε ένα αποτέλεσμα που θα έμοιαζε κάπως έτσι: '1~Jan.' ή '1~January'. Η αλληλουχία λοιπόν πολλών αλφαριθμητικών μπορεί να γίνει με τον ειδικό χαρακτήρα '#'.

Ένα ακόμη χαρακτηριστικό του BibTeX είναι και η υποστήριξη της διασταύρωσης (cross-referencing) 2 καταχωρήσεων. Ας δούμε ένα παράδειγμα για αυτό. Έστω ότι στο έγγραφο (π.χ LaTeX) υπάρχει το \cite{no-gnats} και υποθέτουμε ότι υπάρχουν οι εξής δύο καταχωρήσεις στη βιβλιογραφική βάση:

```
@INPROCEEDINGS(no-gnats,
crossref = "gg-proceedings",
author = "Rocky Gneisser",
title = "No Gnats Are Taken for Granite",
pages = "133-139")
…
@PROCEEDINGS(gg-proceedings,
editor = "Gerald Ford and Jimy Carter",
title = "The Gnats and Gnus 1988 Proceedings",
booktitle = "The Gnats and Gnus 1988 Proceedings")
```

τότε το ειδικό πεδίο crossref λέει στο BibTeX ότι η καταχώρηση με κλειδί το no-gnats θα πρέπει να «κληρονομήσει» (inherit) κάθε πεδίο που του λείπεται από την καταχώρηση με κλειδί το gg-proceedings. Στην περίπτωση αυτή δηλαδή τα δύο πεδία editor και booktitle. Τέλος υποστηρίζεται και η «ειδική εντολή» @PREAMBLE για τη βιβλιογραφική βάση που συντάσεται όπως και η @STRING μόνο που δεν υπάρχει μεταβλητή (όνομα πεδίου) ούτε και το σύμβολο ανάθεσης τιμής '='.π.χ:

```
@PREAMBLE{"\newcommand{\noopsort[1]{}"#"\newcommand{\singleletter}[1]{#1} "}.
```

2.1.5 Παράδειγμα BibTeX

Το BibTeX είναι ένας τύπος αρχείου (*.bib) απλού κειμένου, ανεξαρτήτου στυλ, για τη λίστα των βιβλιογραφικών αντικειμένων. Ας δούμε ένα παραδείγμα BibTeX για την καλύτερη κατανόηση των παραπάνω.

```
% a sample bibliography file


@article{small,
```

```
author = {Freely, I.P.},
title = {A small paper},
journal = {The journal of small papers},
year = 1997,
volume = {1},
note = {to appear},
}

@book{big,
author = {Jass, Hugh},
title = {A big paper},
publisher = {Jack Black},
year = 7991,
volume = {MCMXCVII},
}
```

Παραπάνω έχουμε μια μικρή βιβλιογραφική βάση BibTeX με δύο καταχωρήσεις. Η $1^{η}$ ανήκει στην κατηγορία **article** όπως φαίνεται από την αντίστοιχη εντολή @article που υπάρχει πριν από την είσοδο σε αυτήν την καταχώρηση, και η $2^{η}$ ανήκει στην κατηγορία **book** φαίνεται από την αντίστοιχη εντολή @book. Στη συνέχεια ακολουθεί μια λέξη κλειδί για την κάθε καταχώρηση, το small για την $1^{η}$ και το big για τη $2^{η}$, κι έπειτα τα πεδία με τις αντίστοιχες τιμές τους. Το κλειδί αυτό δεν έχει καμία σχέση με το key που συναντήσαμε παραπάνω στην περιγραφή των πεδίων. Χρησιμοποιείται κυρίως για λόγους ευρετηριοποίησης και διαστάυρωσης των καταχωρήσεων και είναι προαιρετικό. Τέλος η $1^{η}$ γραμμή

% a sample bibliography style

αγνοείται τελείως από το BibTeX καθώς όσες γραμμές ξεκινάνε με πρώτο χαρακήρα το '%' θεωρούνται ως σχόλια.

2.2 Σχετικά Εργαλεία

Για έναν τόσο διαδεδομένο τύπο βιβλιογραφικών βάσεων δεδομένων όπως αυτόν του BibTeX είναι σίγουρο πως υπάρχουν πολλά εργαλεία για την επεξεργασία και τη διαχείριση των εγγραφών αυτών. Υπάρχει σαφώς μια μεγάλη ποικιλία σε χαρακτηριστικά, υποστηριζόμενες λειτουργίες, γλώσσες προγραμματισμού, υποστηριζόμενες πλατφόρμες και σε διάφορα άλλα τεχνικά και προγραμματιστικά χαρακτηριστικά όλων αυτών των έτοιμων

πακέτων, εργαλείων και βιβλιοθηκών. Εμείς θα τα διακρίνουμε κυρίως σε 3 κατηγορίες, σ'αυτά που παρέχουν ένα γραφικό περιβάλλον στο χρήστη (Graphical User Interface – GUI), σ'αυτά που δεν του παρέχουν κάποιο γραφικό περιβάλλον και σε βιβλιοθήκες (libs). Θα αναφέρουμε παρακάτω τα πιο σημαντικά από την κάθε κατηγορία, καθώς και τις λειτουργίες που παρέχουν στο χρήστη, περιληπτικά.

2.2.1 Γραφικά Εργαλεία

Τα γραφικά εργαλεία απευθύνονται πάντα στον τελικό χρήστη. Σκοπός τους είναι η διευκόλυνση του χρήστη στη διαχείριση των βιβλιογραφικών βάσεων, όπως στην εύρεση των εγγραφών στο αρχείο, την προσθήκη μιας νέας εγγραφής σ'αυτό, την επεξεργασία ή διαγραφή εγγραφών από το αρχείο κ.λ.π. Τα εργαλεία αυτά είναι αρκετά φιλικά προς το χρήστη, ο οποίος και δε χρειάζεται να γνωρίζει και πολλά γι'αυτά. Χαρακτηριστικά παραδείγματα εδώ είναι:

- **Allbib**[5]: Πρόκειται για ένα script γραμμένο σε γλώσσα Perl για να διαχειρίζεται βιβλιογραφικές βάσεις BibTeX. Παρέχεται στο χρήστη γραφικό περιβάλλον σε GTK.
- **Aigaion**[6]: Διαδικτυακό σύστημα διαχείρισης βιβλιογραφικών βάσεων. Είναι γραμμένο σε PHP/MYSQL.
- **BibEdit**[7]: Εργαλείο για τη δημιουργία ή επεξεργασία αρχείων BibTeX.
- **Bib-It**[8]: Εφαρμογή γραμμένη σε Java για τη διαχείριση BibTeX αρχείων.
- **BibORB**[9]: Διαδικτυακή εφαρμογή που διαχειρίζεται και διαμοιράζει βιβλιογραφίες BibTeX σε ένα δίκτυο.
- **GBib**[10]: Λογισμικό για τη διαχείριση των εγγραφών βιβλιογραφικών βάσεων BibTeX. Παρέχει γραφικό περιβάλλον σε GTK.
- **JabRef**[11]: Εφαρμογή με γραφικό περιβάλλον σε Java για τη διαχείριση βιβλιογραφικών βάσεων BibTeX κι όχι μόνο.
- **KBibTeX**[12]: Κειμενογράφος βιβλιογραφικών βάσεων BibTeX. Παρέχει γραφικό περιβάλλον σε QT.
- **Pybliographer**[13]: Εργαλείο διαχείρισης βιβλιογραφικών βάσεων.
- **Tkbibtex**[14]: Ένας φορητός κειμενογράφος βιβλιογραφικών βάσεων BibTeX γραμμένος σε Tcl/Tk.

2.2.2 Μη-Γραφικά Εργαλεία

Στην κατηγορία αυτή ανήκει ένα πολύ μεγάλο πλήθος εφαρμογών. Αυτό ισχύει διότι ακόμα και τα περισσότερα από τα παραπάνω ανήκουν και σ'αυτήν την κατηγορία, προσφέροντας τη δυνατότητα στο χρήστη να χρησιμοποιήσει ή όχι γραφικό περιβάλλον. Επιπλέον, κάποια από αυτά τα εργαλεία έρχονται σε μορφή εφαρμογής, δηλ. ένα ή και περισσότερα εκτελέσιμα αρχεία έτοιμα προς εκτέλεση από τη γραμμή εντολών. Όπως τα γραφικά, έτσι κι αυτά απευθύνονται πάντα στον τελικό χρήστη, ο οποίος εδώ πρέπει να γνωρίζει κάποια στοιχειώδη πράγματα για τη λειτουργία τους, αφού είναι λιγότερο φιλικά προς αυτόν. Τα πιο χαρακτηριστικά παραδείγματα εδώ είναι:

- **Bibutils**[15] : Πρόκειται για ένα σύνολο προγραμμάτων που αλληλομετατρέπει διάφορους τύπους βιβλιογραφικών βάσεων χρησιμοποιώντας ως ενδιάμεσο έναν XML τύπο. Στο σύνολο αυτό ανήκουν τα: bib2xml, copac2xml, end2xml, endx2xml, isi2xml, med2xml, modsclean, ris2xml, xml2bib, xml2end, xml2isi, xml2ris, xml2word.
- **Bibliography (BibTeX) Tools**[16]: Μια συλλογή προγραμμάτων που διαχειρίζονται αρχεία βιβλιογραφικών βάσεων σε μορφή BibTeX. Αναφορικά τα προγράμματα που συμπεριλαμβάνονται στη συλλογή αυτή είναι τα: bibcheck, bibclean, bibextract, bibjoin, biblabel, biborder, bibparse, bibsearch, bibsort, bibsplit.

2.2.3 Βιβλιοθήκες

Στην κατηγορία αυτή ανήκουν τα εργαλεία που έρχονται σε μορφή βιβλιοθήκης προσφέροντας μια σειρά από έτοιμες προγραμματιστικές λειτουργίες (συναρτήσεις). Σε αντίθεση με τα γραφικά και μη-γραφικά εργαλεία, οι βιβλιοθήκες απευθύνονται σε προγραμματιστές, οι οποίοι μπορούν να τις ενσωματώσουν στον κώδικά τους προκειμένου να κατασκευάσουν ένα εργαλείο που θα ανήκει τελικά σε μία από τις δύο πιο πάνω κατηγορίες. Χαρακτηριστικό παράδειγμα εδώ είναι το:

- **btOOL**[17]: Η διαπροσωπεία ενός προγραμματιστή σε αρχεία της μορφής BibTeX. Πρόκειται για ένα ζεύγος βιβλιοθηκών, η μία είναι γραμμένη σε γλώσσα προγραμματισμού C και η άλλη σε Perl, που δίνουν στον προγραμματιστή άμεση πρόσβαση σε αρχεία BibTeX. Περιλαμβάνει τη βιλιοθήκη btparse, η οποία είναι γραμμένη σε C, και τη Text::BibTeX που είναι γραμμένη σε Perl.

2.3 Αναγκαιότητα της Πτυχιακής

Αξιοσημείωτο είναι πως παρά το μεγάλο αριθμό εργαλείων για τη διαχείριση βιβλιογραφικών βάσεων BibTeX, δεν υπάρχει κάποιο πακέτο που να είναι πλήρες, να ανταποκρίνεται δηλαδή πλήρως στις απαιτήσεις του χρήστη. Τα περισσότερα από τα προαναφερθέντα εργαλεία έχουν πολλούς περιορισμούς όσον αφορά τα χαρακτηριστικά που υποστηρίζουν και τις λειτουργίες που παρέχουν στον τελικό χρήστη. Πιο συγκεκριμένα:

- Λίγα είναι αυτά που αναγνωρίζουν όλες τις κατηγορίες των καταχωρήσεων του BibTeX, αφού περιορίζονται συνήθως σε καταχωρήσεις που ενδιαφέρουν μόνο στους σχεδιαστές τους, κάνοντας μη δυνατή τη γενική τους χρήση.
- Ένας ακόμη σημαντικός περιορισμός είναι κι αυτός των μακροεντολών με τη χρήση της «ειδικής καταχώρησης» @STRING, οπου δίνεται η δυνατότητα στο χρήστη να ορίζει ένα μεγάλο πλήθος τέτοιων μακροεντολών και να τις διαχειρίζεται όπως θέλει κάθε φορά μέσα στη βιβλιογραφική βάση BibTeX. Εδώ είναι ακόμη λιγότερα αυτά που μπορούν να διαχειρίζονται τις μακροεντολές αυτές και να επιτρέπουν στο χρήστη να τροποποιεί τις τιμές των αλφαριθμητικών που περιέχουν τέτοιες μεταβλητές.
- Επίσης το χαρακτηριστικό της διασταύρωσης δύο καταχωρήσεων με τον ειδικό τύπο πεδίου crossref είναι κάτι που δεν το συναντήσαμε σε κανένα εργαλείο.
- Τέλος δεν υπάρχει η δυνατότητα για τροποποίηση του μέγιστου αριθμού πεδίων που μπορεί να περιέχει μια καταχώρηση, καθώς και το μέγιστο πλήθος χαρακτήρων που μπορεί να έχει ένα αλφαριθμητικό.

Αυτοί είναι αρκετά σημαντικοί περιορισμοί που, επειδή δεν ισχύουν στη δική μας εργασία, την κάνουν πολύ πιο ευέλικτη από τις άλλες. Βέβαια ο τρόπος υλοποίησης της βιβλιοθήκης μας έχει κι αυτός έναν μικρό περιορισμό: δε δίνει τη δυνατότητα στο χρήστη να επεξεργάζεται πολλά αρχεία BibTeX συγχρόνως, αλλά ένα μόνο αρχείο τη φορά, κάνοντας λίγο πιο χρονοβόρα τη διαδικασία επεξεργασίας πληροφοριών διαφορετικών βιβλιογραφικών βάσεων BibTeX.

3

Ανάλυση και Σχεδίαση

Σ'αυτό το κεφάλαιο παρουσιάζεται αναλυτικά ο σχεδιασμός και η υλοποίηση του λεκτικού και συντακτικού αναλυτή που φτιάξαμε για την αναγνώριση των εγγραφών των αρχείων BibTeX. Επίσης δίνονται και οι δομές δεδομένων που χρησιμοποιούνται για την αποθήκευση των πληροφοριών αυτών των εγγραφών.

3.1 Αναγνώριση Εγγραφών

Εδώ θα δείξουμε τον τρόπο με τον οποίο αναγνωρίζονται οι καταχωρήσεις στην βιβλιογραφική βάση. Θα δούμε τα βήματα υλοποίησης του λεκτικού αναλυτή και τη γλώσσα που αναγνωρίζει αυτός. Επίσης περιγράφεται η γραμματική που αναγνωρίζει ο συντακικός μας αναλυτής.

3.1.1 Λεκτική Ανάλυση

Ο λεκτικός αναλυτής πραγματοποιείται σε 3 ξεχωριστά επίπεδα:

- Επίπεδο κορυφής. (Επίπεδο 1)
- Μέσα σε καταχώρηση. (Επίπεδο 2)

- Αλφαριθμητικό. (Επίπεδο 3)

Το επίπεδο κορυφής είναι το αρχικό επίπεδο. Μπαίνουμε στο 2ο επίπεδο όταν συναντήσουμε ένα '@' στο 1ο επίπεδο, και όταν συναντήσουμε ένα '}' ή ')' που τελειώνει μια καταχώρηση επιστρέφουμε στο 1ο επίπεδο. Μπαίνουμε στο 3ο επίπεδο μόλις συναντήσουμε ένα ' " ', ή μόλις συναντήσουμε ένα '{' που να μη σηματοδοτεί όμως την έναρξη μιας καταχώρησης. Η γλώσσα του λεκτικού αναλυτή δεν είναι κανονική (γιατί τα '{' και '}' πρέπει να ισορροπούν) και είναι γλώσσα με συμφραζόμενα (γιατί το '{' μπορεί να σημαίνει κάτι διαφορετικό κάθε φορά ανάλογα με τη γραμματική έκφραση). Έτσι θα χρησιμοποιήσουμε κανονικές εκφράσεις για να περιγράψουμε τα λεκτικά στοιχεία, γιατί είναι το σημείο εκκίνησης του ίδιου του λεκτικού αναλυτή.

3.1.1.1 Επίπεδο κορυφής

Στο επίπεδο κορυφής αναγνωρίζονται τα παρακάτω σύμβολα σύμφωνα με τις κανονικές εκφράσεις στα δεξιά:

```
AT          \@
NEWLINE     \n
COMMENT     \%~[\n]*\n
WHITESPACE  [\ \r\t]+
JUNK        ~[\@\n\ \r\t]+
```

3.1.1.2 Μέσα σε καταχώρηση

Συναντώντας το σύμβολο AT (@) στο επίπεδο κορυφής εισερχόμαστε σε αυτό το επίπεδο. Εδώ αναγνωρίζονται τα παρακάτω σύμβολα:

```
NEWLINE     \n
COMMENT     \%~[\n]*\n
WHITESPACE  [\ \r\t]+
NUMBER      [0-9]+
NAME        [a-z0-9\!\$\&\*\+\-\.\/\:\;\<\>\?\[\]\^\_\`\|]+
LBRACE      \{
RBRACE      \}
LPAREN      \(
RPAREN      \)
EQUALS      =
```

```
HASH        \#
COMMA ,
QUOTE \"
```

Βρισκόμαστε στο επίπεδο αυτό μέχρι να αναγνωρίσουμε το σύμβολο RBRACE ή RPAREN που θα δηλώνει το τέλος μιας καταχώρησης, κι όχι οποιοδήποτε RBRACE ή RPAREN σύμβολο. Τότε ξαναγυρίζουμε στο παραπάνω επίπεδο.

3.1.1.3 Αλφαριθμητικό

Στο επίπεδο αυτό εισερχόμαστε κάθε φορά που αναγνωρίζουμε το σύμβολο QUOTE ή το σύμβολο LBRACE που δεν σηματοδοτεί την έναρξη κάποιας καταχώρησης. Οπότε φεύγουμε από αυτό το επίπεδο όταν συναντήσουμε αντίστοιχα το σύμβολο QUOTE ή το σύμβολο RBRACE. Πρέπει επίσης να σημειωθεί ότι μέσα σε ένα αλφαριθμητικό μπορεί να περιέχεται ο χαρακτήρας ‘ “ ‘ ή ο χαρακτήρας ‘ { ‘ , αρκεί βεβαίως να έχει προηγηθεί το ειδικό σύμβολο ‘ \ ‘.

3.1.1.4 Παράδειγμα

Ας δούμε ένα παράδειγμα για το τι αναγνωρίζει ο λεκτικός αναλύτής σε κάθε σημείο του αρχείου (§2.1.5 – Αριστερά φαίνεται ο αριθμός της κάθε γραμμής):

```
1  % a sample bibliography file
2
3  @article{small,
4  author = {Freely, I.P.},
5  title = {A small paper},
6  journal = {The journal of small papers},
7  year = 1997,
8  volume = {1},
9  note = {to appear},
10 }
11
12 @book{big,
13 author = {Jass, Hugh},
14 title = {A big paper},
```

```
15 publisher = {Jack Black},
16 year = 7991,
17 volume = {MCMXCVII},
18 }
```

Από τη γραμμή 1 μέχρι την 3 ο λεκτικός αναλυτής βρίσκεται στο επίπεδο 1. Διαβάζει ένα COMMENT (γραμμή 1) και WHITESPACE's μέχρι να φτάσει στη γραμμή 3 και να διαβάσει το χαρακτήρα '@' οπότε και μπαίνει στο επίπεδο 2 με το σύμβολο AT. Εκεί διαβάζει ένα NAME (article), ένα LBRACE ({), ένα NAME (small) κι ένα COMMA (,). Στη γραμμή 4 διαβάζει NAME (author), EQUALS (=), LBRACE ({) οπότε και μπαίνει στο επίπεδο 3 και διαβάζει ένα αλφαρηθμητικό ({Freely, I.P.}) και διαβάζοντας το '}' επιστρέφει στο επίπεδο 2 και διαβάζει το COMMA (,). Στις γραμμές 5-9 γίνεται η ίδια ακριβώς διαδικασία, εκτος της 7 όπου δε μπαίνει στο επίπεδο 2, αλλά διαβάζει ένα NUMBER (1997). Στη γραμμή 10 διαβάζει ένα RBRACE (}), οπότε και επιστρέφει στο επίπεδο 1. Συνεχίζει με την ίδια διαδικασία και στις άλλες καταχωρήσεις μέχρι να εξαντληθούν όλες.

3.1.2 Συντακτική Ανάλυση

Η συντακτική ανάλυση έγινε με τη μέθοδο της αναδρομικής κατάβασης χρησιμοποιώντας μια LL_1[(1)] **[34]** γραμματική. Η γλώσσα που χρησιμοποιείται για τη γραμματική είναι σε EBNF **[31-33]** (Extended Backus-Naur Form) μορφή.

Γραμματική:

bibfile : (entry)*

entry : **AT NAME** body

body : **STRING**

| **ENTRY_OPEN** contents **ENTRY_CLOSE**

contents : (**NAME** | **NUMBER**) **COMMA** fields

| fields

---

(1) Γενικά ένας LL συντακτικός αναλυτής (parser) λέγεται $LL_{(k)}$ αν πρέπει να ελέγχει μπροστά k αναγνωριστικά (tokens), κατά της σύνταξη μιας πρότασης, για να πάρει τη σωστή απόφαση.

| value

fields : field { **COMMA** fields }

| ε

field : **NAME EQUALS** value

value : simple_value (**HASH** simple_value)*

simple_value : **STRING**

| **NUMBER**

| **NAME**

Σε αυτήν τη γραμματική, τα τερματικά αντιπροσωπεύονται από λέξεις με κεφαλαία έντονα γράμματα, ενώ τα μη-τερματικά από λέξεις με μικρά γράμματα. (foo)* σημαίνει καμία ή περισσότερες επαναλήψεις του προϊόντος foo, ενώ {foo} σημαίνει ένα προαιρετικό foo. Στην περίπτωση του κανόνα body τα τερματικά ENTRY_OPEN και ENRTY_CLOSE μπορεί να είναι είτε { και } ή (και) , αναλόγως τι αναγνωρίζεται κατά την έναρξη κάποιας καταχώρησης.

3.2 Αποθήκευση Εγγραφών

Μετά τον λεκτικό και τον συντακτικό αναλυτή, είμαστε πλέον σε θέση να αναγνωρίζουμε τις καταχωρήσεις, οπότε τίθεται το ερώτημα που θα αποθηκευτούν αυτές οι πληροφορίες. Πρέπει λοιπόν να δημιουργηθούν κάποιες κατάλληλες δομές δεδομένων για την αποθήκευση αυτών των πληροφοριών. Δημιουργήθηκαν δύο βασικές δομές δεδομένων: μία για την αποθήκευση της πληροφορίας ενός πεδίου, και μία για την αποθήκευση των πληροφοριών μιας ολόκληρης καταχώρησης.

Για την αποθήκευση της πληροφορίας ενός **πεδίου** δημιουργήθηκε η εξής δομή δεδομένων:

```
struct field
{
      int   fid;
      char  *value;
      char  fieldname[MAXNAME];
      char  *expandedvalue;
};
typedef     struct field      field_t;
```

Το fid παίρνει μια δεκαδική τιμή ανάλογα με τον τύπο του πεδίου. Το value έχει σε μορφή αλφαριθμητικού την τιμή αυτού του πεδίου. Το fieldname έχει σε μορφή αλφαριθμητικού το όνομα του πεδίου. Τέλος το expandedvalue έχει σε μορφή αλφαριθμητικού την τιμή του

πεδίου ή την τιμή του πεδίου μετά από την αντικατάσταση των τυχών μεταβλητών που έχουν προκαθοριστεί με κάποια μακροεντολή (αν έχει ζητήσει ο χρήστης μια τέτοια διαδικασία).

Για την αποθήκευση των πληροφοριών μιας **καταχώρησης** δημιουργήθηκε η εξής δομή δεδομένων:

```
struct entry
{
      int         type;
      char        *key;
      field_t     fields[MAXIMUM_FIELDS];
      int         nf;                     /* Num of fields */
      struct entry      *next;
};
typedef struct entry    entry_t;
typedef entry_t   *Entry_t;
```

Το type παίρνει μια δεκαδική τιμή ανάλογα με τον τύπο της καταχώρησης. Το key παίρνει σε μορφή αλφαριθμητικού την τιμή του κλειδιού της καταχώρησης ή το κενό αλφαριθμητικό (“”) αν δεν υπάρχει κλειδί. Το nf είναι ένας δεκαδικός αριθμός που δείχνει το πλήθος των πεδίων της συγκεκριμένης καταχώρησης. Τέλος το fields είναι ένας πίνακας με τα πεδία της καταχώρησης και το next είναι ο δείκτης στον επόμενο κόμβο.

Και στις 2 δομές δεδομένων οι μεταβλητές με τα κεφαλαία γράμματα είναι σταθερές που έχουν οριστεί με κάποια μακροεντολή.

3.3 Επέκταση Εγγραφών

Όπως έχουμε πει, το BibTeX υποστηρίζει τον ορισμό μακροεντολών με τον ειδικό τύπο καταχώρησης @STRING. Στη συνέχεια, καθεμιά από τις μακροεντολές αυτές μπορεί να χρησιμοποιηθεί στην τιμή οποιουδήποτε πεδίου οποιασδήποτε καταχώρησης, κάνοντας έτσι πιο εύκολη τη δημιουργία καταχωρήσεων που περιέχουν πεδία με την ίδια τιμή (ή ένα κομμάτι αυτής). Π.χ:

```
@STRING(WGA = “World Gnus Almanac”)
@BOOK(almanac-66,
title = 1966 # WGA,
…)
@BOOK(almanac-67,
title = 1967 # WGA,
…) κ.ο.κ.
```

Κατά την αναγνώριση και αποθήκευση των εγγραφών αυτών, στην τιμή του πεδίου title αποθηκεύεται η τιμή “1966 # WGA” (αντίστοιχα για το 67) ως έχει. Μετά το τέλος της

αποθήκευσης, δίνεται η δυνατότητα στο χρήστη-προγραμματιστή να επεκτείνει την τιμή των πεδίων που περιέχουν μακροεντολές. Στη συγκεκριμένη περίπτωση, μετά την επέκταση, το πεδίο title θα έχει τιμή "1966 World Gnus Almanac" (αντίστοιχα για το 67). Αυτό γίνεται ως εξής: Κρατάμε σε μια δομή δεδομένων τις διάφορες μακροεντολές που συναντάμε στο αρχείο BibTeX. Η δομή αυτή έχει τη μορφή

```
Struct sl
{
      char sname[MAXNAME];
      char *svalue;
      struct sl *next;
};
typedef struct sl sl_t;
typedef sl_t *SL_t;
```

Το sname έχει σε μορφή αλφαριθμητικού το όνομα της μακροεντολής, το svalue έχει σε μορφή αλφαριθμητικού την πραγματική τιμή αυτής της μακροεντολής και το next είναι ο δείκτης στον κόμβο με την επόμενη μακροεντολή. Έτσι έχοντας όλες τις εγγραφές αποθηκευμένες σε μια δομή δεδομένων και τις μακροεντολές σε μια άλλη, ελέγχουμε για κάθε μια μακροεντολή αν υπάρχει αυτή στην τιμή ενός (ή και παραπάνω) πεδίου οποιασδήποτε εγγραφής και την αντικαθιστούμε με την πραγματική της τιμή.

4

Υλοποίηση

4.1 Λεπτομέρειες Υλοποίησης

Σκοπός της πτυχιακής εργασίας ήταν η δημιουργία μιας βιβλιοθήκης που θα παρέχει στο χρήστη-προγραμματιστή ένα API (Application Programming Interface) για την επεξεργασία βιβλιογραφικών βάσεων BibTeX. Το API αυτό λοιπόν αποτελείται από κάποιες βασικές συναρτήσεις, οι οποίες εκτελούν κάποιες βασικές λειτουργίες πάνω στη βιβλιογραφική βάση. Χωρίζεται σε 2 κατηγορίες, API υψηλού επιπέδου (High-Level API) και API χαμηλού επιπέδου (Low-Level API). Ας δούμε λοιπόν μία μία τις συναρτήσεις αυτές και τι κάνει η κάθε μία.

- `void MBP_dbinit(void);`
- `int MBP_dbopen(char *filename);`
- `void MBP_dbclose(void);`

Η συνάρτηση **MBP_dbinit** αρχικοποιεί τη βιβλιοθήκη **mbp**. Αρχικοποιεί κατάλληλα δηλαδή κάποιες καθολικές μεταβλητές για την ομαλή λειτουργία της. Καλείται μια φορά πριν από τη χρήση οποιασδήποτε συνάρτησης.

Η συνάρτηση **MBP_dbopen** ανοίγει το BibTeX αρχείο που ορίζεται με το filename και αρχικοποιεί τον καθολικό δείκτη αρχείου στο αρχείο αυτό. Καλείται μια φορά μετά την **MBP_dbinit** και πριν από οποιαδήποτε άλλη συνάρτηση. Μετά την κλήση της, η λειτουργία

οποιασδήποτε συνάρτησης εκτελείται πάνω στο συγκεκριμένο αρχείο. Σε περίπτωση επιτυχίας επιστρέφει την τιμή 0. Σε περίπτωση αποτυχίας επιστρέφει -1.

Η συνάρτηση **MBP_dbclose** κλείνει το τρέχον BibTeX αρχείο επεξεργασίας και ελευθερώνει όλη τη μνήμη που κατέλαβε η βιβλιοθήκη κατά την αρχικοποίηση. Σε περίπτωση που ο χρήστης-προγραμματιστής θέλει να επεξεργαστεί κάποιο άλλο αρχείο BibTeX θα πρέπει να καλέσει πρώτα αυτήν την συνάρτηση και μετά να ξανακαλέσει τις **MBP_dbinit** και **MBP_dbopen** με όρισμα το νέο αρχείο στη $2^{η}$. Αυτό θα πρέπει να επαναλαμβάνεται κάθε φορά που πρέπει να γίνει αλλαγή αρχείου.

4.1.1 API Υψηλού Επιπέδου

Οι συναρτήσεις της κατηγορίας αυτής είναι:

- `int MBP_AllRecords(int *nor, Entry_t *list, SL_t *userstrings);`
- `int MBP_FindKeyedRecords(int *nor, Entry_t *list, char *key);`
- `int MBP_FindRecords(int *nor, Entry_t *list, char *searchstring);`
- `int MBP_ExpandStrings(Entry_t *E,SL_t S);`

Η συνάρτηση **MBP_AllRecords** ελέγχει πρώτα αν το αρχείο BibTeX έχει ήδη αναλυθεί συντακτικά. Αν όχι, καλεί το συντακτικό αναλυτή ο οποίος και επεξεργάζεται το αρχείο BibTeX και αναγνωρίζει τις καταχωρήσεις του. Στη συνέχεια αποθηκεύει όλες τις καταχωρήσεις σε μια καθολική λίστα τύπου Entry_t. Η λίστα αυτή χρησιμοποιείται στη συνέχεια από τη βιβλιοθήκη κάθε φορά που πρέπει να εκτελεστεί μια λειτουργία πάνω στις καταχωρήσεις. Όσο είναι «ανοιχτή» η βιβλιοθήκη μπορεί να επεξεργάζεται μόνο αυτό το αρχείο. Σε περίπτωση επιτυχίας του συντακτικού αναλυτή η καθολική μεταβλητή **mbperrno** παίρνει την τιμή **NOERROR**. Σε περίπτωση αποτυχίας η καθολική μεταβλητή **mbperrno** έχει πάρει μια από τις τιμές που δείχνουν το λόγω του σφάλματος. Οι τιμές αυτές μπορεί να είναι:

- **MBP_EPARSE**. Αυτό σημαίνει ότι έγινε κάποιο λάθος κατά τη συντακτική ανάλυση του αρχείου, δηλαδή υπάρχει κάποιο συντακτικό λάθος στο BibTeX αρχείο.
- **MBP_EMEM**. Αυτό σημαίνει πως δεν υπήρχε αρκετή μνήμη στο σύστημα για αυτή τη διαδικασία.
- **MBP_EFILE**. Αυτό σημαίνει πως δημιουργήθηκε κάποιο σφάλμα κατά τη μετακίνηση του δείκτη μέσα στο BibTeX αρχείο. Δηλαδή έγινε κάποιο λάθος κατά την κλήση της *fseek* ή της *ftell* ή κάτι τέτοιο.

- **MBP_ESEGM**. Αυτό σημαίνει πως πήγε να γίνει εισαγωγή ενός πεδίου σε κενή καταχώρηση, δηλαδή σε NULL δείκτη. Επίσης μπορεί να σημαίνει ότι σε μια καταχώρηση πήγε να γίνει εισαγωγή ενός αριθμού πεδίων μεγαλύτερο από το επιτρεπόμενο.
- **MBP_ECROSS**. Αυτό σημαίνει ότι πήγε να γίνει διαστάυρωση (cross-referencing) μιας καταχώρησης με μια μη υπάρχουσα καταχώρηση, δηλαδή κλειδί που δεν ανήκει σε καμιά καταχώρηση.

Αν το BibTeX αρχείο έχει ήδη αναλυθεί συντακτικά, τότε επεξεργάζεται την καθολική λίστα που δημιουργείται μετά την κλήση του συντακτικού αναλυτή και αποθηκεύει όλες τις κανονικές καταχωρήσεις (όχι @STRING) στη λίστα list και όλες τις καταχωρήσεις της μορφής @STRING (δηλ. τις μακροεντολές) στη λίστα userstrings. Επιπλέον στο nor αποθηκεύεται το πλήθος των κανονικών καταχωρήσεων που βρέθηκαν στο αρχείο BibTeX. Είναι σημαντικό τα list και userstrings να έχουν αρχικοποιηθεί στο NULL πριν από κάθε κλήση της συνάρτησης, διαφορετικά τα αποτελέσματα μπορεί να μην είναι τα επιθυμητά. Η συνάρτηση **MBP_AllRecords** επιστρέφει -1 σε περίπτωση αποτυχίας ή έναν μη αρνητικό ακέραιο σε περίπτωση επιτυχίας (ή 0 αν δε βρεθεί καμία κανονική καταχώρηση).

Η συνάρτηση **MBP_FindKeyedRecords** ελέγχει αν το BibTeX αρχείο έχει ήδη αναλυθεί συντακτικά, κι αν δεν έχει αναλυθεί καλεί το συντακτικό αναλυτή (όπως και η **MBP_AllRecords**) . Στη συνέχεια ψάχνει στην καθολική λίστα, που έχει δημιουργηθεί κατά την συντακτική ανάλυση, για να βρει την καταχώρηση που έχει κλειδί το key και την αποθηκεύει στο list. Στο nor αποθηκεύεται το πλήθος των καταχωρήσεων που βρέθηκαν. Αυτό κανονικά θα πρέπει να είναι 1 σε περίπτωση επιτυχίας και 0 διαφορετικά. Κι εδώ είναι σημαντικό το list να έχει αρχικοποιηθεί στο NULL πριν από την κάθε κλήση της συνάρτησης. Σε πείπτωση αποτυχίας επιστρέφει -1 ή έναν μη αρνητικό ακέραιο σε περίπτωση επιτυχίας (ή 0 αν δε βρεθεί καμία κανονική καταχώρηση).

Η συνάρτηση **MBP_FindRecords** δουλεύει με τον ίδιο τρόπο που δουλεύει και η **FindKeyedRecords** με τη διαφορά ότι δεν ψάχνει μόνο τα κλειδιά των καταχωρήσεων, αλλά και τις τιμές των πεδίων της κάθε καταχώρησης για να βρει το searchstring.

Η συνάρτηση **MBP_ExpandStrings** επεκτείνει όλα τα αλφαριθμητικά σύμβολα από όλες τις τιμές των πεδίων όλων των καταχωρήσεων της λίστας E με τις κατάλληλες τιμές από αυτές που βρίσκονται στη λίστα αλφαριθμητικών (μακροεντολών) S. Τη λίστα S μπορούμε να την πάρουμε από την κλήση της **MBP_AllRecords** μέσω του userstrings. Μετά την κλήση αυτής της συνάρτησης, το πεδίο expandedvaule στην πληροφορία των πεδίων των καταχωρήσεων της λίστας E θα έχει τη νέα τιμή, η οποία θα είναι της μορφής {…} ανεξάρτητα από το ποια ήταν η αρχική τιμή. Επιστρέφει το πλήθος των επεκτάσεων που έγιναν ή 0 αν δεν έγινε καμία.

4.1.2 API Χαμηλού Επιπέδου

Οι συναρτήσεις της κατηγορίας αυτής είναι:

- `void MBP_AddRecord(Entry_t *list, Entry_t E);`
- `Entry_t MBP_dbnext(void);`

Η συνάρτηση **MBP_AddRecord** προσθέτει την καταχώρηση E στο τέλος της λίστας καταχωρήσεων list.

Η συνάρτηση **MBP_dbnext** επιστρέφει την επόμενη καταχώρηση του αρχείου BibTeX ανεξαρτήτου τύπου. Αυτό που ουσιαστικά γίνεται είναι να καλεί το συντακτικό αναλυτή μέχρι να αναγνωρισθεί μία μόνο καταχώρηση, την οποία και επιστρέφει, αφήνοντας κάθε φορά τον καθολικό δείκτη αρχείου στη θέση που έμεινε μετά την τελευταία κλήση αυτής της συνάρτησης. Σε περίπτωση σφάλματος επιστρέφει το NULL και η καθολική μεταβλητή **mbperrno** παίρνει την αντίστοιχη τιμή.

Όλες αυτές τις συναρτήσεις (μαζί κι αυτές του υψηλού επιπέδου) λοιπόν τις ενσωματώσαμε σε μια βιβλιοθήκη, την **mbp** η οποία αποτελούσε και το στόχο της πτυχιακής εργασίας.

4.2 Πλατφόρμες και Προγραμματιστικά Εργαλεία

Η υλοποίηση της βιβλιοθήκης έγινε σε γλώσσα προγραμματισμού C. Αποτελείται από 17 αρχεία που περιέχουν περίπου 2000 γραμμές πηγαίου κώδικα, συν 1 αρχείο Makefile, (βλ.Παράρτημα Α § Α.1). Επομένως για να μπορέσει να τη χρησιμοποιήσει ο χρήστης-προγραμματιστής απαιτούνται ορισμένα εργαλεία. Πρώτα απ'όλα για τη μεταγλώττιση του πηγαίου κώδικα και τη δημιουργία της βιβλιοθήκης **mbp**, δηλαδή το **libmbp.a** απαιτείται ένας μεταγλωττιστής της C. Συνιστάται ο μεταγλωττιστής gcc (GNU C Compiler) με τον οποίο κι έχει δοκιμαστεί και δουλεύει άψογα. Επίσης απαιτείται κι ένα εργαλείο για την εκτέλεση του Makefile που δίνεται στον πηγαίο κώδικα. Εμείς χρησιμοποιήσαμε το πιο διαδεδομένο εργαλείο για αυτή τη δουλειά, το make. Τέλος, για την ενσωμάτωση όλων των object αρχείων, που προκύπτουν κατά τη μεταγλώττιση του πηγαίου κώδικα, σε ένα αρχείο (libmbp.a) χρησιμοποιήθηκε το εργαλείο ar. Η όλη ανάπτυξη και εκτέλεση της εφαρμογής (της βιβλιοθήκης) έγιναν σε λειτουργικό σύστημα Linux, ένα από τα πιο φιλικά, ασφαλή και

πιο ενδεδειγμένα για προγραμματιστικούς σκοπούς λειτουργικά συστήματα, στο οποίο και απευθύνεται κυρίως (προεπιλογή). Φυσικά είναι δυνατή και η τροποποίηση του κώδικα προκειμένου να μπορεί να χρησιμοποιηθεί και από άλλα λειτουργικά συστήματα. Συγκεκριμένες απαιτήσεις σε υλικό υπολογιστή δεν υπάρχουν.

5

Χρήση και Συμπεράσματα

5.1 Τρόπος Χρήσης

Στην ενότητα αυτή δίνονται κάποια παραδείγματα για τον τρόπο χρήσης της βιβλιοθήκης **libmbp.a**, το πως μπορεί δηλαδή ο χρήστης-προγραμματιστής να την ενσωματώσει στον κώδικα του και να καλέσει κάποια από τις συναρτήσεις που δίνονται στο API.

Παράδειγμα 1^{ο} : Επεξεργασία ενός μόνο αρχείου BibTeX

```
#include <mbp/mbp.h>

int main(void)
{
 Entry_t E;
 SL_t S;
 Int x;

 MBP_dbinit();
 MBP_dbopen("myfile.bib");
 E = NULL;
```

```
 S = NULL;
 MBP_AllRecords(&x,&E,&S);
 … Process the Data …
 MBP_dbclose();
 return 0;
}
```

Παράδειγμα 2ο : Επεξεργασία δύο αρχείων BibTeX

```
#include <mbp/mbp.h>

int main(void)
{
 Entry_t E1, E2;
 SL_t S1, S2;
 Int x1, x2;

 MBP_dbinit();
 MBP_dbopen("file1.bib");
 E1 = NULL;
 S1 = NULL;
 MBP_AllRecords(&x1,&E1,&S1);
 MBP_ExpandStrings(&E1,S1);
 … Process the Expanded Data  of the file1.bib …
 MBP_dbclose();

 MBP_dbinit();
 MBP_dbopen("file2.bib");
 E2 = NULL;
 MBP_FindRecords(&x2,&E2,"logic");
 … Process the Data  of the file2.bib …
 MBP_dbclose();
 return 0;
}
```

5.2 Συμπεράσματα

Συνοψίζοντας, μπορούμε να πούμε πως φτιάξαμε μια βιβλιοθήκη (**libmbp.a**) και το API της που αποτελείται από συναρτήσεις, οι οποίες εκτελούν κάποιες βασικές λειτουργίες για την αναγνώριση των καταχωρήσεων μιας βιβλιογραφικής βάσης BibTeX. Πιο συγκεκριμένα, αυτό γίνεται σε 4 στάδια:

- ✓ Συντακτική ανάλυση της βιβλιογραφικής βάσης BibTeX.
- ✓ Αναγνώριση των καταχωρήσεων της βιβλιογραφικής βάσης BibTeX.
- ✓ Εύρεση συγκεκριμένων καταχωρήσεων με βάση κάποιο κλειδί αναζήτησης.
- ✓ Αποθήκευση των καταχωρήσεων αυτών σε μια κατάλληλη δομή δεδομένων για περαιτέρω επεξεργασία από τον χρήστη-προγραμματιστή.

Πρόκειται για μια αρκετά ευέλικτη βιβλιοθήκη με ελάχιστους περιορισμούς και πολλά πλεονεκτήματα σε σχέση με άλλα εργαλεία για την επεξεργασία βιβλιογραφικών βάσεων BibTeX. Σίγουρα είναι η καλύτερη δυνατή επιλογή για τον προγραμματιστή που θέλει να υλοποιήσει μια εφαρμογή για την επεξεργασία βιβλιογραφικών βάσεων BibTeX.

6

Βιβλιογραφία

[1] Oren Patashnik. BibTeXing, 8 February 1988.

[2] LaTeX
http://en.wikipedia.org/wiki/LaTeX

[3] BibTeX
http://en.wikipedia.org/wiki/BibTeX

[4] TeX
http://en.wikipedia.org/wiki/TeX

[5] Allbib
http://allbib.sourceforge.net/

[6] Aigaion
http://www.aigaion.nl/

[7] BibEdit
http://www.iui.se/staff/jonasb/bibedit/

[8] Bib-It
http://bib-it.sourceforge.net/

[9] BibORB
http://biborb.glymn.net/

[10] Gbib

http://gbib.seul.org/

[11] JabRef

http://jabref.sourceforge.net/

[12] KBibTeX

http://www.unix-ag.uni-kl.de/~fischer/kbibtex/

[13] Pybliographer

http://www.pybliographer.org/

[14] Tkbibtex

http://www.cat.csiro.au/ict/staff/pic/tkbibtex.html

[15] Bibutils

http://www.scripps.edu/~cdputnam/software/bibutils/

[16] Bibliography (BibTeX) Tools

http://www.ecst.csuchico.edu/~jacobsd/bib/tools/bibtex.html

[17] btOOL

http://www.gerg.ca/software/btOOL/

[18] Mittelbach, Frank; Goosens, Michel (2004). The LaTeX Companion, 2nd edition, Addison-Wesley. ISBN 0-201-36299-6

[19] Lamport, Leslie (1994). LaTeX: A document preparation system: User's guide and reference, illustrations by Duane Bibby, 2nd edition, Reading, Mass: Addison-Wesley Professional. ISBN 0-201-52983-1

[20] Kopka, Helmut; Daly, Patrick W. (2003). Guide to LaTeX, 4th edition, Addison-Wesley Professional. ISBN 0-321-17385-6

[21] Griffiths, David F.; Highman, David S. (1997). Learning LaTeX. Philadelphia: Society for Industrial and Applied Mathematics. ISBN 0-898-71383-8

[22] Donald E. Knuth. The TeXbook (Computers and Typesetting, Volume A). Reading, Massachusetts: Addison-Wesley, 1984. ISBN 0-201-13448-9

[23] Donald E. Knuth. TeX: The Program (Computers and Typesetting, Volume B). Reading, Massachusetts: Addison-Wesley, 1986. ISBN 0-201-13437-3

[24] Donald E. Knuth. Digital Typography (CSLI lecture notes, no 78). Center for the Study of Language and Information, 1999. ISBN 1-57586-010-4

[25] Donald E. Knuth and Michael F. Plass. Breaking Paragraphs Into Lines, Software — Practice and Experience 11 (1981), 1119–1184. Reprinted as chapter 3 of Digital Typography, p. 67–155

[26] Leslie Lamport. LaTeX: A Document Preparation System. Addison-Wesley, Reading, Massachusetts: Addison-Wesley, 1986. ISBN 0-201-52983-1

[27] Franklin Mark Liang. Word Hy-phen-a-tion by Com-put-er, PhD thesis, Department of Computer Science, Stanford University, August 1983

[28] M.D. Spivak. The Joy of TeX (2nd edition). American Mathematical Society, 1990. ISBN 0-8218-2997-1

[29] Nelson H.F. Beebe. 25 Years of TeX and METAFONT: Looking Back and Looking Forward, TUGboat 25 (2004), 7–30

[30] Michael Vulis, Modern TeX and Its Applications, CRC Press, 1992. ISBN 0-8493-4431-X

[31] EBNF
http://en.wikipedia.org/wiki/Extended_Backus-Naur_form

[32] Niklaus Wirth: What can we do about the unnecessary diversity of notation for syntactic definitions? CACM, Vol. 20, Issue 11, November 1977, pp. 822-823

[33] Roger S. Scowen: Extended BNF — A generic base standard. Software Engineering Standards Symposium 1993

[34] LL Parsers
http://en.wikipedia.org/wiki/LL%281%29

ΠΑΡΑΡΤΗΜΑ Α

Α.1 Πηγαίος Κώδικας

Ο πηγαίος κώδικας αποτελείται από 17 αρχεία (περίπου 2000 γραμμές κώδικα). Περιληπτικά τα αρχεία αυτά είναι:

- **api.h**. Αρχείο κεφαλίδα με τα προτότυπα των συναρτήσεων του API.
- **api.c**. Αρχείο με τον κώδικα των συναρτήσεων του API.
- **common.h**. Αρχείο κεφαλίδα με τα προτότυπα κάποιων βοηθητικών συναρτήσεων.
- **common.c**. Αρχείο με τον κώδικα κάποιων βοηθητικών συναρτήσεων.
- **conf.h**. Αρχείο κεφαλίδα με τον ορισμό κάποιων εξωτερικών μεταβλητών που έχουν να κάνουν με τη συμπεριφορά της βιβλιοθήκης.
- **error.h**. Αρχείο κεφαλίδα με τα προτότυπα συναρτήσεων που έχουν να κάνουν με τον έλεγχο ορθότητας των αρχείων BibTeX.
- **error.c**. Αρχείο με τον κώδικα συναρτήσεων που έχουν να κάνουν με τον έλεγχο ορθότητας των αρχείων BibTeX.
- **global.h**. Αρχείο κεφαλίδα με τη δήλωση των καθολικών μεταβλητών της βιβλιοθήκης.
- **lex.h.** Αρχείο κεφαλίδα με τα προτότυπα των συναρτήσεων που χρησιμοποιούνται για τη λεκτική ανάλυση των αρχείων BibTeX. Επίσης ορίζονται και τα σύμβολα που αναγνωρίζει ο λεκτικός αναλυτής.
- **lex.c**. Αρχείο με τον κώδικα των συναρτήσεων που χρησιμοποιούνται για τη λεκτική ανάλυση των αρχείων BibTeX, δηλαδή η υλοποίηση του λεκτικού αναλυτή.
- **mbp.h**. Το κύριο αρχείο κεφαλίδα της βιβλιοθήκης.

- **parser.h**. Αρχείο κεφαλίδα με τα προτότυπα συναρτήσεων που χρησιμοποιούνται για τη συντακτική ανάλυση των αρχείων BibTeX.
- **parser.c**. Αρχείο με τον κώδικα των συναρτήσεων που χρησιμοποιούνται για τη συντακτική ανάλυση των αρχείων BibTeX, δηλαδή η υλοποίηση του συντακτικού αναλυτή.
- **string.h**. Αρχείο κεφαλίδα με τα προτότυπα συναρτήσεων που έχουν να κάνουν με την επεξεργασία αλφαριθμητικών.
- **string.c**. Αρχείο με τον κώδικα συναρτήσεων που έχουν να κάνουν με την επεξεργασία αλφαριθμητικών.
- **symbol.h**. Αρχείο κεφαλίδα με τα προτότυπα συναρτήσεων που έχουν να κάνουν με την επεξεργασία των δομών δεδομένων με τις πληροφορίες των καταχωρήσεων των αρχείων BibTeX. Εδώ ορίζονται και οι δομές αυτές στις οποίες και αποθηκεύονται οι καταχωρήσεις.
- **symbol.c**. Αρχείο με τον κώδικα συναρτήσεων που έχουν να κάνουν με την επεξεργασία των δομών δεδομένων με τις πληροφορίες των καταχωρήσεων των αρχείων BibTeX.

A.1.1 api.h

```
#if defined(MBP_H)
#ifndef API_H
#define API_H



/*****************************************/
/*****         HIGH LEVEL            *****/
/*****************************************/



/* This function parses the bibtex file and stores its records to *list
 * and its string records {e.g @STRING} to *userstrings. It also stores the number of
 * records found to nor.On success it returns a non-negative number, or 0 if no record
found. If
 * any error occurs during parsing it returns -1.
 * Caution: list should be set to NULL before calling this function or the new records
 *          will be added to the existing ones. Same for userstrings too.
 */
int MBP_AllRecords(int *nor, Entry_t *list, SL_t *userstrings);




/* This function parses the bibtex file and searches for records with
 * with key (char * key). nor is the number of records found. Normally this would be 1
 * on success or 0 if none is found. If any error occurs during parsing it returns -1.
 * Caution: list should be set to NULL before calling this function or the new records
 *          will be added to the existing ones.
 */
int MBP_FindKeyedRecords(int *nor, Entry_t *list, char *key);
```

```
/* This function does what FindKeyedRecords with the difference that not only searches
 * all the record keys, but field values as well.
 * Caution: list should b set to NULL before calling this function or the new records
 *          will be added to the existing ones.
 */
int MBP_FindRecords(int *nor, Entry_t *list, char *searchstring);



/* This function expands all the strings [{....} or "...."] of all records E with the
 * appropriate values found in S. It returns the number of expansions made or 0 if
none made.
 * The expanded values will be available through (*E)->fields[i].expandedvalue
 * After this call the expanded value field will be a string {...} no matter what was
the last value
 */
int MBP_ExpandStrings(Entry_t *E,SL_t S);



/* This function adds the record E to the record list list*/
void MBP_AddRecord(Entry_t *list, Entry_t E);




/* This function initializes the mbp library. Call this function once before using any
of the
 * rest api functions.
 */
void MBP_dbinit(void);


/*  This function opens the specified bibtex file.On success it returns 0, -1
otherwise. */
int MBP_dbopen(char *filename);


/* This function frees the memory allocated by the mbp library and closes the bibtex
file
 * currently being processed. Once it is called no api function is available.
 */
void MBP_dbclose(void);




/*****************************************/
/*****         LOW LEVEL              *****/
/*****************************************/


/* This function "returns" the next entry of the bibfile. */
Entry_t MBP_dbnext(void);



#endif
#endif
```

A.1.2 api.c

```
#include "mbp.h"
```

```
/*****************************************/
/*****         API FUNCTIONS          *****/
/*****************************************/




int MBP_AllRecords(int *nor, Entry_t *list, SL_t *userstrings)
{
        Entry_t tmp;
        int i, counter=0;


        if(!isparsed)
        {
                rewind(fp);
                Parser();
        }
        if(mbperrno != NOERROR) return -1;

        tmp = EntryList;
        for(EntryList=EntryList; EntryList!=NULL; EntryList=EntryList->next)
        {
                if(EntryList->type == STRING_E)
                {
                        for(i=0; i<EntryList->nf; i++)
                                AddString(userstrings,EntryList-
>fields[i].fieldname,EntryList->fields[i].value);
                        continue;
                }
                MBP_AddRecord(list,EntryList);
                counter++;
        }
        EntryList = tmp;
        *nor = counter;


        return  counter;
}



int MBP_FindKeyedRecords(int *nor, Entry_t *list, char *key)
{
        Entry_t tmp;
        char *keydup, *rkeydup;
        int counter=0;

        if(!isparsed)
        {
                rewind(fp);
                Parser();
        }
        if(mbperrno != NOERROR) return -1;

        keydup = Strdup(key);
        Lowercase(&keydup);

        tmp = EntryList;
        for(EntryList=EntryList; EntryList!=NULL; EntryList=EntryList->next)
        {
                rkeydup = Strdup(EntryList->key);
                Lowercase(&rkeydup);

                if(!strcmp(rkeydup,keydup))
                {
                        MBP_AddRecord(list,EntryList);
                        counter++;
                }

                free(rkeydup);
        }
        EntryList = tmp;
        free(keydup);
```

```
        *nor = counter;


        return counter;
}



int MBP_FindRecords(int *nor, Entry_t *list, char *searchstring)
{
        Entry_t tmp;
        char *stringdup, *searchdup;
        int counter=0, flag=0, i;

        if(!isparsed)
        {
                rewind(fp);
                Parser();
        }
        if(mbperrno != NOERROR) return -1;

        searchdup = Strdup(searchstring);
        Lowercase(&searchdup);

        tmp = EntryList;
        for(EntryList=EntryList; EntryList!=NULL; EntryList=EntryList->next)
        {
                stringdup = Strdup(EntryList->key);
                Lowercase(&stringdup);
                if(strstr(stringdup,searchdup) != NULL)
                {
                        MBP_AddRecord(list,EntryList);
                        counter++;
                        free(stringdup);
                        continue;
                }
                free(stringdup);


                for(i=0; i<EntryList->nf; i++)
                {
                        flag = 0;
                        stringdup = Strdup(EntryList->fields[i].fieldname);
                        Lowercase(&stringdup);
                        if(strstr(stringdup,searchdup) != NULL)
                        {
                                MBP_AddRecord(list,EntryList);
                                counter++;
                                flag = 1;
                                free(stringdup);
                                break;
                        }
                        free(stringdup);


                        stringdup = Strdup(EntryList->fields[i].value);
                        Lowercase(&stringdup);
                        if(strstr(stringdup,searchdup) != NULL)
                        {
                                MBP_AddRecord(list,EntryList);
                                counter++;
                                flag = 1;
                                free(stringdup);
                                break;
                        }
                        free(stringdup);


                        if(EntryList->fields[i].expandedvalue == NULL)       continue;
                        stringdup = Strdup(EntryList->fields[i].expandedvalue);
                        Lowercase(&stringdup);
                        if(strstr(stringdup,searchdup) != NULL)
                        {
                                MBP_AddRecord(list,EntryList);
                                counter++;
                                flag = 1;
```

```
                                free(stringdup);
                                break;
                        }
                        free(stringdup);
                }

                if(flag == 1)  continue;
        }
        EntryList = tmp;
        free(searchdup);

        *nor = counter;


        return counter;
}


int MBP_ExpandStrings(Entry_t *E, SL_t S)
{
        Entry_t E_tmp;
        int i, expanded=0;


        if(S == NULL)  return expanded;       /* No @STRING entries */

        E_tmp = *E;
        for(*E=*E; *E!=NULL; *E=(*E)->next)
        {
                if((*E)->type == PREAMBLE  ||  (*E)->type == STRING_E)
                        continue;

                for(i=0; i<(*E)->nf; i++)
                        expanded += ExpandString(&((*E)->fields[i].expandedvalue),S);
        }
        *E = E_tmp;

        return expanded;
}




void MBP_AddRecord(Entry_t *list, Entry_t E)
{
        Entry_t tmp;
        int i;


        if(E == NULL)  return;

        if(*list == NULL)
        {
                *list = (Entry_t)malloc(sizeof(entry_t));
                if(*list == NULL)      SetError(MBP_EMEM);
                (*list)->type = E->type;
                (*list)->key = Strdup(E->key);
                for(i=0; i<E->nf; i++)
                {
                        (*list)->fields[i].fid = E->fields[i].fid;
                        (*list)->fields[i].value = Strdup(E->fields[i].value);
                        strcpy((*list)->fields[i].fieldname,E->fields[i].fieldname);
                        (*list)->fields[i].expandedvalue           =           Strdup(E-
>fields[i].expandedvalue);
                }
                (*list)->nf = E->nf;
                (*list)->next = NULL;

                return;
        }


        tmp = *list;
        for(*list=*list; (*list)->next!=NULL; *list=(*list)->next);

        (*list)->next = (Entry_t)malloc(sizeof(entry_t));
```

```
        if((*list)->next == NULL)     SetError(MBP_EMEM);
        (*list)->next->type = E->type;
        (*list)->next->key = Strdup(E->key);
        for(i=0; i<E->nf; i++)
        {
                (*list)->next->fields[i].fid = E->fields[i].fid;
                (*list)->next->fields[i].value = Strdup(E->fields[i].value);
                strcpy((*list)->next->fields[i].fieldname,E->fields[i].fieldname);
                (*list)->next->fields[i].expandedvalue           =           Strdup(E-
>fields[i].expandedvalue);
        }
        (*list)->next->nf = E->nf;
        (*list)->next->next = NULL;
        *list = tmp;
}



void MBP_dbinit(void)
{
        Initialize();
}


int MBP_dbopen(char *filename)
{
        fp = fopen(filename,"r");
        if(fp == NULL)
        {
                mbperrno = MBP_EFILE;
                return -1;
        }
        return 0;
}



void MBP_dbclose(void)
{
        if(fclose(fp)) mbperrno = MBP_EFILE;

        FreeBibMem();
}



Entry_t MBP_dbnext(void)
{
        int i, foundentry, jmpval;
        Entry_t NE;


        if((jmpval=setjmp(env)) == 0)
        {
                Lex();
                if(TOKEN_ID == AT  &&  LEVEL == MODE2)
                {
                        if(fseek(fp,-strlen(TOKEN),SEEK_CUR)           ==           -1)
        SetError(MBP_EFILE);
                        foundentry = entry();
                        if(foundentry == FALSE)
                                return MBP_dbnext();

                        if(EntryList != NULL)
                        {
                                NE = (Entry_t)malloc(sizeof(entry_t));
                                if(NE == NULL) SetError(MBP_EMEM);
                                NE->type = EntryList->type;
                                NE->key = Strdup(EntryList->key);
                                for(i=0; i<EntryList->nf; i++)
                                {
                                        NE->fields[i].fid = EntryList->fields[i].fid;
                                        NE->fields[i].value      =      Strdup(EntryList-
>fields[i].value);
                                        strcpy(NE->fields[i].fieldname,EntryList-
>fields[i].fieldname);
```

```
                                        NE->fields[i].expandedvalue  =  Strdup(EntryList-
>fields[i].expandedvalue);
                                }
                                NE->nf = EntryList->nf;
                                NE->next = NULL;

                                free(EntryList);
                                EntryList = NULL;

                                mbperrno = NOERROR;
                                return NE;
                        }
                }
        }

        return NULL;
}
```

A.1.3 common.h

```
#if defined(MBP_H)
#ifndef COMMON_H
#define COMMON_H

#define FALSE          0
#define TRUE           1


void Initialize(void);
void FreeBibMem(void);

#endif
#endif
```

A.1.4 common.c

```
#include "mbp.h"



void Initialize(void)
{
        if((TOKEN = (char *)malloc(MAXNAME * sizeof(char))) == NULL)
                SetError(MBP_EMEM);
        TOKEN_ID = SPACE;
        LINES = 1;
        LEVEL = MODE1;
        COMMENT_FLAG = 0;
        isparsed = FALSE;
        EntryList = NULL;
        StringList = NULL;

        NOMF[ARTICLE-1] =      4;
        NOMF[BOOK-1] =         4;
        NOMF[BOOKLET-1] =      1;
        NOMF[CONFERENCE-1] =   4;
        NOMF[INBOOK-1] =       5;
        NOMF[INCOLLECTION-1] = 5;
        NOMF[INPROCEEDINGS-1] =        4;
        NOMF[MANUAL-1] =       1;
        NOMF[MASTERSTHESIS-1] =        4;
        NOMF[MISC-1] =         0;
        NOMF[PHDTHESIS-1] =    4;
        NOMF[PROCEEDINGS-1] =  2;
        NOMF[TECHREPORT-1] =   4;
        NOMF[UNPUBLISHED-1] =  3;
        NOMF[STRING_E-1] =     0;
        NOMF[PREAMBLE-1] =     0;

}
```

```
void FreeBibMem(void)
{
        free(TOKEN);
        DeleteEntries(&EntryList);
        DeleteStrings(&StringList);
}
```

A.1.5 conf.h

```
#if defined(MBP_H)
#ifndef CONF_H
#define CONF_H

#define ENABLE_CROSSREF                             1       /* Set to 1 enable cross
referencing.0 otherwise */
#define ENABLE_WARNINGS                             1       /* Set to 1 to print
warning on screen if a required field is missing from an entry. 0 otherwise */
#define MAXIMUM_LENGTH_OF_NAME_OR_NUMBER     128     /* Don't set too high unless
needed*/
#define MAXIMUM_NUMBER_OF_FIELDS_PER_RECORD  100     /* Don't set too high unless
needed*/
#define STRING_CHUNK                         128     /* Extend memory allocated for
STRINGS by STRING_CHUNK bytes. DO NO SET LESS THAN MAXIMUM_LENGTH_OF_NAME_OR_NUMBER */


#endif
#endif
```

A.1.6 error.h

```
#define NOERROR         0
#define MBP_EPARSE      1
#define MBP_EMEM        2
#define MBP_EFILE       3
#define MBP_ESEGM       4
#define MBP_ECROSS      5

int mbperrno;
short NOMF[16];               /* 16 is the number of entry types */
                      /* NOMF stands for: Number Of Minimum Fields */

void SetError(int error);
void CrossRef(void);
Entry_t FindCrossEntry(Entry_t E, char *key);
void CrossEntries(Entry_t *cross, int i, Entry_t source);
void CheckFields(void);
int RequiredField(int entry_type, int field_id);
```

A.1.7 error.c

```
#include "mbp.h"



void SetError(int error)
{
        mbperrno = error;
        longjmp(env,error);
}



void CrossRef(void)
```

```
{
        Entry_t start, source;
        char *buf;
        int i;

        start = EntryList;
        for(EntryList=EntryList; EntryList!=NULL; EntryList=EntryList->next)
        {
                for(i=0; i<EntryList->nf; i++)
                {
                        buf = Strdup(EntryList->fields[i].fieldname);
                        Lowercase(&buf);
                        if(!strcmp(buf,"crossref"))
                        {
                                source          =          FindCrossEntry(start,EntryList-
>fields[i].value);
                                if(source == NULL) SetError(MBP_ECROSS);
                                CrossEntries(&EntryList,i,source);
                                free(buf);
                                break;
                        }
                        free(buf);
                }
        }
        EntryList = start;
}




Entry_t FindCrossEntry(Entry_t E, char *key)
{
        Entry_t start, ret;
        char *buf, *keybuf;


        keybuf = Strdup(key);
        Lowercase(&keybuf);

        start = E;
        for(E=E; E!=NULL; E=E->next)
        {
                buf = Strdup(E->key);
                Lowercase(&buf);

                if(!strcmp(buf,keybuf))
                {
                        free(buf);
                        free(keybuf);
                        ret = E;
                        E = start;

                        return ret;
                }
                free(buf);
        }
        E = start;
        free(keybuf);

        return NULL;
}



void CrossEntries(Entry_t *cross, int i, Entry_t source)
{
        int j, k, found, i_isfilled=0;


        for(j=0; j<source->nf; j++)
        {
                found = 0;
                for(k=0; k<(*cross)->nf; k++)
                        if(source->fields[j].fid == (*cross)->fields[k].fid)
                                found++;
```

```
                if(found == 0)
                {
                        if(!i_isfilled)
                        {
                                (*cross)->fields[i].fid = source->fields[j].fid;
                                (*cross)->fields[i].value        =        Strdup(source-
>fields[j].value);
                                (*cross)->fields[i].expandedvalue    =    Strdup(source-
>fields[j].expandedvalue);
                                strcpy((*cross)->fields[i].fieldname,source-
>fields[j].fieldname);
                                i_isfilled = 1;
                        }

                        else
                        {
                                (*cross)->fields[(*cross)->nf].fid       =       source-
>fields[j].fid;
                                (*cross)->fields[(*cross)->nf].value   =   Strdup(source-
>fields[j].value);
                                (*cross)->fields[(*cross)->nf].expandedvalue            =
Strdup(source->fields[j].expandedvalue);
                                strcpy((*cross)->fields[(*cross)->nf].fieldname,source-
>fields[j].fieldname);
                                (*cross)->nf += 1;
                        }
                }
        }
}




void CheckFields(void)
{
        Entry_t tmp;
        int i, foundrequired, counter=0;

        tmp = EntryList;
        for(EntryList=EntryList; EntryList!=NULL; EntryList=EntryList->next)
        {
                counter++;
                if(EntryList->type==MISC  ||  EntryList->type==STRING_E  ||  EntryList-
>type==PREAMBLE)
                        continue;

                foundrequired = 0;
                for(i=0; i<EntryList->nf; i++)
                        if(RequiredField(EntryList->type,EntryList->fields[i].fid))
foundrequired++;
                if(foundrequired < NOMF[EntryList->type-1])
                        fprintf(stderr,"***WARNING: Required field(s) missing in record
No%d\tRecordKey: %s\n",counter,EntryList->key);
        }
        EntryList = tmp;
}


int RequiredField(int entry_type, int field_id)
{
        if(entry_type == ARTICLE)
                if(field_id!=AUTHOR   &&   field_id!=TITLE   &&   field_id!=JOURNAL   &&
field_id!=YEAR)
                        return FALSE;

        if(entry_type == BOOK)
                if(field_id!=AUTHOR   &&   field_id!=EDITOR   &&   field_id!=TITLE   &&
field_id!=PUBLISHER && field_id!=YEAR)
                        return FALSE;

        if(entry_type == BOOKLET)
                if(field_id != TITLE)
                        return FALSE;

        if(entry_type == CONFERENCE  ||  entry_type == INPROCEEDINGS)
```

```
                if(field_id!=AUTHOR   &&   field_id!=TITLE   &&   field_id!=BOOKTITLE   &&
field_id!=YEAR)
                        return FALSE;

        if(entry_type == INBOOK)
                if(field_id!=AUTHOR   &&   field_id!=EDITOR   &&   field_id!=TITLE   &&
field_id!=PAGES && field_id!=PUBLISHER && field_id!=YEAR)
                        return FALSE;

        if(entry_type == INCOLLECTION)
                if(field_id!=AUTHOR   &&   field_id!=TITLE   &&   field_id!=BOOKTITLE   &&
field_id!=PUBLISHER && field_id!=YEAR)
                        return FALSE;

        if(entry_type == MANUAL)
                if(field_id!=TITLE)
                        return FALSE;

        if(entry_type == MASTERSTHESIS)
                if(field_id!=AUTHOR   &&   field_id!=TITLE   &&   field_id!=SCHOOL   &&
field_id!=YEAR)
                        return FALSE;

        if(entry_type == PHDTHESIS)
                if(field_id!=AUTHOR   &&   field_id!=TITLE   &&   field_id!=SCHOOL   &&
field_id!=YEAR)
                        return FALSE;

        if(entry_type == PROCEEDINGS)
                if(field_id!=TITLE && field_id!=YEAR)
                        return FALSE;

        if(entry_type == TECHREPORT)
                if(field_id!=AUTHOR  &&  field_id!=TITLE  &&  field_id!=INSTITUTION  &&
field_id!=YEAR)
                        return FALSE;

        if(entry_type == UNPUBLISHED)
                if(field_id!=AUTHOR && field_id!=TITLE && field_id!=NOTE)
                        return FALSE;

        return TRUE;
}
```

A.1.8 global.h

```
#if defined(MBP_H)
#ifndef GLOBAL_H
#define GLOBAL_H

FILE    *fp;
Entry_t EntryList;
SL_t    StringList;
char    *TOKEN;
int     TOKEN_ID, LINES, LEVEL, COMMENT_FLAG, et, fid, isparsed;
jmp_buf env;

#endif
#endif
```

A.1.9 lex.h

```
#if defined(MBP_H)
#ifndef LEX_H
#define LEX_H

#define SPACE           0       /* ' ' */
#define NUMBER          1       /* (digit)+ */
#define NAME            2       /* (NAMECHAR)+ */
```

```
#define STRING          3       /* "....."  ||  {.....}  [including \" or \{ as a char]
*/
#define AT              4       /* @ */
#define LBRACE          5       /* { */
#define RBRACE          6       /* } */
#define LPAREN          7       /* ( */
#define RPAREN          8       /* ) */
#define EQUALS          9       /* = */
#define HASH            10      /* # */
#define COMMA           11      /* , */
#define QUOTE           12      /* " */

#define MOD             13      /* % */
#define NEWLINE         14      /* \n */
#define  END_OF_FILE    15      /* EOF */



/* NAMECHARS */
/* a-z, A-Z characters and 0-9 digits included */
#define EXCLAMATION     16      /* ! */
#define SCALAR          17      /* $ */
#define AMPERSAND       18      /* & */
#define MUL             19      /* * */
#define PLUS            20      /* + */
#define MINUS           21      /* - */
#define DOT             22      /* . */
#define SLASH           23      /* / */
#define COLON           24      /* : */
#define SEMICOLON       25      /* ; */
#define LT              26      /* < */
#define GT              27      /* > */
#define QUESTION        28      /* ? */
#define LAGGYLH         29      /* [ */
#define RAGGYLH         30      /* ] */
#define HAT             31      /* ^ */
#define UNDERSCORE      32      /* _ */
#define QUOTATION       33      /* ' */
#define PIPE            34      /* | */



#define COMMENT_ERROR   -1      /* EOF before NewLine  ('\n') */
#define GENERAL_ERROR   -2      /* We'll talk about that later */


#define MODE1           1       /* Top Level */
#define MODE2           2       /* In-Entry Level */
#define MODE3           3       /* String Level */



int IsNameChar(char c);
int IsNumber(char *s);
void Lex(void);

#endif
#endif
```

A.1.10 lex.c

```
#include "mbp.h"


int IsNameChar(char c)
{
        if(isalpha(c)) return TRUE;
        if(isdigit(c)) return TRUE;
        if(c == '!')   return TRUE;
        if(c == '$')   return TRUE;
        if(c == '&')   return TRUE;
```

```
        if(c == '*')   return TRUE;
        if(c == '+')   return TRUE;
        if(c == '-')   return TRUE;
        if(c == '.')   return TRUE;
        if(c == '/')   return TRUE;
        if(c == ':')   return TRUE;
        if(c == ';')   return TRUE;
        if(c == '<')   return TRUE;
        if(c == '>')   return TRUE;
        if(c == '?')   return TRUE;
        if(c == '[')   return TRUE;
        if(c == ']')   return TRUE;
        if(c == '^')   return TRUE;
        if(c == '_')   return TRUE;
        if(c == '\'')  return TRUE;
        if(c == '|')   return TRUE;

        return FALSE;
}



int IsNumber(char *s)
{
        int i;
        for(i=0; *(s+i)!='\0'; i++)
                if( !isdigit(*(s+i)) )
                        return FALSE;
        return TRUE;
}




void Lex(void)
{
        char c, temp;
        int i, ENTRY_counter, chunk_counter;


        i = 0;
        chunk_counter = 1;

        if(LEVEL == MODE1)     /* Top-Level MODE */
        {
                while(isspace(c = fgetc(fp)))
                        if(c == '\n') LINES++;

                if(c == EOF)
                {
                        strcpy(TOKEN,"");
                        TOKEN_ID = END_OF_FILE;
                        return;
                }


                if(c == '%')
                {
                        while((c = fgetc(fp)) != '\n')
                                if(c == EOF)
                                {
                                        fprintf(stderr,"WARNING:  Unfinished  comment  at
line %d\n",LINES);
                                        TOKEN_ID = END_OF_FILE;
                                        return;
                                }

                        LINES++;
                        Lex();

                        return;
                }


                if(c == '@')
                {
```

```
                strcpy(TOKEN,"@");
                TOKEN_ID = AT;
                LEVEL = MODE2;
                return;
        }


        while((c = fgetc(fp)) != '\n')               /* Read JUNKS */
                if(c == EOF)
                {
                        TOKEN_ID = END_OF_FILE;
                        return;
                }
        Lex();

        return;
}



if(LEVEL == MODE2)     /* In-Entry-Level MODE */
{
        while(isspace(c = fgetc(fp)))
                if(c == '\n') LINES++;

        if(c == '%')
        {
                COMMENT_FLAG = 1;

                while((c = fgetc(fp)) != '\n')
                        if(c == EOF)   SetError(MBP_EPARSE);

                LINES++;
                Lex();

                return;
        }


        if(IsNameChar(c))
        {
                while(IsNameChar(c))
                {
                        *(TOKEN+i) = c;
                        i++;
                        c = fgetc(fp);
                }
                *(TOKEN+i) = '\0';

                TOKEN_ID = NAME;
                if(IsNumber(TOKEN)) TOKEN_ID = NUMBER;

                ungetc(c,fp);
                return;
        }


        if(c == '{')
        {
                if(COMMENT_FLAG == 1)
                {
                        COMMENT_FLAG = LBRACE;
                        LEVEL = MODE3;
                        Lex();
                        return;
                }

                strcpy(TOKEN,"{");
                TOKEN_ID = LBRACE;
                COMMENT_FLAG = 1;

                return;
        }

        if(c == '}')
        {
                strcpy(TOKEN,"}");
```

```
                TOKEN_ID = RBRACE;
                COMMENT_FLAG = 0;
                LEVEL = MODE1;          /* End of this entry */
                return;
        }

        if(c == '(')
        {
                strcpy(TOKEN,"(");
                TOKEN_ID = LPAREN;
                return;
        }

        if(c == ')')
        {
                strcpy(TOKEN,")");
                TOKEN_ID = RPAREN;
                COMMENT_FLAG = 0;
                LEVEL = MODE1;          /* End of this entry */
                return;
        }

        if(c == '=')
        {
                strcpy(TOKEN,"=");
                TOKEN_ID = EQUALS;
                return;
        }

        if(c == '#')
        {
                strcpy(TOKEN,"#");
                TOKEN_ID = HASH;
                return;
        }

        if(c == ',')
        {
                strcpy(TOKEN,",");
                TOKEN_ID = COMMA;
                return;
        }

        if(c == '\"')
        {
                LEVEL = MODE3;
                COMMENT_FLAG = QUOTE;
                Lex();                  /* String found */
                return;
        }


        if(c == EOF)    SetError(MBP_EPARSE);
}


if(LEVEL == MODE3)      /* String-Level MODE */
{
        ENTRY_counter = 0;

        if(COMMENT_FLAG == LBRACE)
        {
                temp = '{';
                 *(TOKEN+i) = '{';
                i++;

                while((c  =  fgetc(fp))  !=  '}'    ||    temp  ==  '\\'    ||
ENTRY_counter != 0)
                {
                        if(c == '\n') LINES++;
                        if(c == EOF)   SetError(MBP_EPARSE);
                        if(c == '{'  &&  temp != '\\') ENTRY_counter++;
                        if(c == '}'  &&  temp != '\\') ENTRY_counter--;

                        if(i >= (chunk_counter*STRING_CHUNK) - 1)
                        {
```

```
                                        chunk_counter++;
                                        TOKEN       =       (char       *)realloc((char
*)TOKEN,(chunk_counter*STRING_CHUNK)*sizeof(char));
                                        if(TOKEN == NULL) SetError(MBP_EMEM);
                                }

                                temp = c;
                                *(TOKEN+i) = c;
                                i++;
                        }
                        *(TOKEN+i) = '}';
                        *(TOKEN+i+1) = '\0';
                }


                if(COMMENT_FLAG == QUOTE)
                {
                        temp = '\"';
                         *(TOKEN+i) = '\"';
                        i++;

                        while((c = fgetc(fp)) != '\"'  ||  temp == '\\')
                        {
                                if(c == '\n') LINES++;
                                if(c == EOF)   SetError(MBP_EPARSE);

                                if(i >= (chunk_counter*STRING_CHUNK) - 1)
                                {
                                        chunk_counter++;
                                        TOKEN       =       (char       *)realloc((char
*)TOKEN,(chunk_counter*STRING_CHUNK)*sizeof(char));
                                        if(TOKEN == NULL) SetError(MBP_EMEM);
                                }

                                temp = c;
                                *(TOKEN+i) = c;
                                i++;
                        }
                        *(TOKEN+i) = '\"';
                        *(TOKEN+i+1) = '\0';
                }

                TOKEN_ID = STRING;
                LEVEL = MODE2;          /* End of String. Return to previous Level (In-
Entry) */
                COMMENT_FLAG = 1;
        }
}
```

A.1.11 mbp.h

```
#ifndef MBP_H
#define MBP_H

/* Standar headers */
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <setjmp.h>


/* Additional Headers */
#include "conf.h"
#include "common.h"
#include "lex.h"
#include "parser.h"
#include "symbol.h"
#include "string.h"
#include "global.h"
#include "api.h"
#include "error.h"
```

```
#endif
```

A.1.12 parser.h

```
#if defined(MBP_H)
#ifndef PARSER_H
#define PARSER_H

void Parser(void);
void bibfile(void);
int entry(void);
void body(void);
void contents(void);
void fields(void);
void field(void);
void value(char **v);
void simple_value(char **v);

#endif
#endif
```

A.1.13 parser.c

```
#include "mbp.h"



void Parser(void)
{
        int jmpval;


        if((jmpval=setjmp(env)) == 0)
        {
                //Initialize();
                bibfile();
                rewind(fp);

                if(ENABLE_CROSSREF)    CrossRef();
                if(ENABLE_WARNINGS)    CheckFields();
                mbperrno = NOERROR;
                isparsed = TRUE;
        }
}



void bibfile(void)
{
        Lex();
        while(TOKEN_ID == AT  &&  LEVEL == MODE2)
        {
                if(fseek(fp,-strlen(TOKEN),SEEK_CUR) == -1)  SetError(MBP_EFILE);
                entry();
                Lex();
        }

}







int entry(void)
{
        Lex();
        if(TOKEN_ID == AT)
        {
                Lex();
```

```
                if(TOKEN_ID == NAME)
                {
                        et = Name2EntryType(TOKEN);
                        if(et == UNKNOWN)             /* Omit unknown entries */
                        {
                                LEVEL = MODE1;
                                return FALSE;
                        }
                        body();
                        return TRUE;
                }

                else
                {
                        SetError(MBP_EPARSE);
                        return FALSE;
                }
        }

        else
        {
                SetError(MBP_EPARSE);
                return FALSE;
        }
}



void body(void)
{
        Lex();

        if(TOKEN_ID == STRING) return;

        if(TOKEN_ID == LBRACE)
        {
                contents();
                LEVEL = MODE2;
                Lex();
                if(TOKEN_ID != RBRACE)
                        SetError(MBP_EPARSE);
                return;
        }

        if(TOKEN_ID == LPAREN)
        {
                contents();
                LEVEL = MODE2;
                Lex();
                if(TOKEN_ID != RPAREN)
                        SetError(MBP_EPARSE);
                return;
        }

        SetError(MBP_EPARSE);
}



void contents(void)
{
        char    *tk;
        long fposition;

        tk = (char *)malloc(MAXNAME*sizeof(char));
        if(tk == NULL) SetError(MBP_EMEM);

        strcpy(tk,"");

        if((fposition = ftell(fp)) == -1) SetError(MBP_EFILE);
        Lex();

        if(TOKEN_ID == STRING)
        {
                if(fseek(fp,-strlen(TOKEN),SEEK_CUR) == -1) SetError(MBP_EFILE);
                AddEntry(&EntryList,et,"");
                value(&tk);
```

```
                InsertField(&EntryList,UNKNOWN,tk,"");
                free(tk);
                return;
        }

        if(TOKEN_ID == NUMBER)
        {
                free(tk);
                tk = Strdup(TOKEN);
                Lex();
                if(TOKEN_ID == COMMA)
                {
                        AddEntry(&EntryList,et,tk);
                        fields();
                        free(tk);
                        return;
                }

                if(fseek(fp,fposition,SEEK_SET) == -1) SetError(MBP_EFILE);
                AddEntry(&EntryList,et,"");
                strcpy(tk,"");
                value(&tk);
                InsertField(&EntryList,UNKNOWN,tk,"");
                free(tk);
                return;
        }

        if(TOKEN_ID == NAME)
        {
                free(tk);
                tk = Strdup(TOKEN);
                Lex();
                if(TOKEN_ID == COMMA)
                {
                        AddEntry(&EntryList,et,tk);
                        fields();
                        free(tk);
                        return;
                }

                if(TOKEN_ID == EQUALS)
                {
                        if(fseek(fp,fposition,SEEK_SET) == -1) SetError(MBP_EFILE);
                        AddEntry(&EntryList,et,"");
                        fields();
                        free(tk);
                        return;
                }

                if(fseek(fp,fposition,SEEK_SET) == -1) SetError(MBP_EFILE);
                AddEntry(&EntryList,et,"");
                strcpy(tk,"");
                value(&tk);
                InsertField(&EntryList,UNKNOWN,tk,"");
                free(tk);
                return;
        }

        if(fseek(fp,-strlen(TOKEN),SEEK_CUR) == -1) SetError(MBP_EPARSE);
        fields();
        free(tk);
}



void fields(void)
{
        Lex();
        if(TOKEN_ID == NAME)
        {
                if(fseek(fp,-strlen(TOKEN),SEEK_CUR) == -1) SetError(MBP_EFILE);
                field();

                Lex();

                if(TOKEN_ID == COMMA)
                        fields();
```

```
                else
                {
                        if(fseek(fp,-strlen(TOKEN),SEEK_CUR) == -1) SetError(MBP_EFILE);
                        return;
                }
        }

        else
                if(fseek(fp,-strlen(TOKEN),SEEK_CUR) == -1) SetError(MBP_EFILE);

}



void field(void)
{
        char    *tk, *v;
        int fid;

        tk = (char *)malloc(MAXNAME*sizeof(char));
        v = (char *)malloc(MAXNAME*sizeof(char));

        if(tk == NULL  ||  v == NULL) SetError(MBP_EMEM);

        strcpy(v,"");

        Lex();
        if(TOKEN_ID == NAME)
        {
                free(tk);
                tk = Strdup(TOKEN);
                fid = Name2FieldId(tk);
                Lex();
                if(TOKEN_ID == EQUALS)
                {
                        value(&v);
                        InsertField(&EntryList,fid,v,tk);
                        free(tk);
                        free(v);
                        return;
                }

                else
                {
                        free(tk);
                        free(v);
                        SetError(MBP_EPARSE);
                }
        }

        else
        {
                free(tk);
                free(v);
                SetError(MBP_EPARSE);
        }
}


void value(char **v)
{
        int PreviousLevel;

        PreviousLevel = LEVEL;

        do
        {
                simple_value(v);
                Lex();
        }
        while(TOKEN_ID == HASH);
        if(fseek(fp,-strlen(TOKEN),SEEK_CUR) == -1) SetError(MBP_EFILE);
        LEVEL = PreviousLevel;
}


void simple_value(char **v)
```

```
{
        char *tmp, *vold;

        Lex();
        if(TOKEN_ID != STRING  &&  TOKEN_ID != NUMBER  &&  TOKEN_ID != NAME)
                SetError(MBP_EPARSE);

        tmp = (char *)malloc(2*sizeof(char));
        if(tmp == NULL)        SetError(MBP_EMEM);

        *tmp = '#';
        *(tmp+1) = '\0';

        vold = Strdup(*v);
        free(*v);
        *v = (char *)malloc((strlen(vold)+strlen(TOKEN)+2)*sizeof(char));
        if(*v == NULL) SetError(MBP_EMEM);
        strcpy(*v,vold);

        strcat(*v,TOKEN);
        strcat(*v,tmp);
}
```

A.1.14 string.h

```
#if defined(MBP_H)
#ifndef STRING_H
#define STRING_H


#define SPECIAL_CHAR   EOF


struct sl
{
        char sname[MAXNAME];
        char *svalue;
        struct sl *next;
};

typedef struct sl sl_t;
typedef sl_t *SL_t;



int Lowercase(char **s);
int AddString(SL_t *S, char *sname, char *svalue);
char *Strdup(const char *s);
int ExpandString(char **s, SL_t S);
void StripBibString(char *source, char **dest);
void DeleteStrings(SL_t *S);
/*void PrintStrings(SL_t S);*/

#endif
#endif
```

A.1.15 string.c

```
#include "mbp.h"


int Lowercase(char **s)
{
        int i;

        for(i=0; *(*s+i)!='\0'; i++)
                *(*s+i) = tolower(*(*s+i));
        return 0;
}
```

```
int AddString(SL_t *S, char *sname, char *svalue)
{
        SL_t temp, NS;

        if(*S == NULL)
        {
                *S = (SL_t)malloc(sizeof(sl_t));
                if(*S == NULL) SetError(MBP_EMEM);
                strcpy((*S)->sname,sname);
                (*S)->svalue = Strdup(svalue);
                (*S)->next = NULL;

                return 0;
        }

        NS = (SL_t)malloc(sizeof(sl_t));
        if(NS == NULL) SetError(MBP_EMEM);
        strcpy(NS->sname,sname);
        NS->svalue = Strdup(svalue);
        NS->next = NULL;

        temp = *S;
        for(*S=*S; (*S)->next!=NULL; *S=(*S)->next);
        (*S)->next = NS;
        *S = temp;

        return 0;
}



char *Strdup(const char *s)
{
        char *ret;

        ret = strdup(s);
        if(ret == NULL)        SetError(MBP_EMEM);
        return ret;
}


int ExpandString(char **s, SL_t S)
{
        SL_t sltmp;
        char *strtoken, *start, *buf, *tmp, previouschar;
        int i, counter=0, opened, chunks;

        strtoken = (char *)malloc(STRING_CHUNK*sizeof(char));
        buf = (char *)malloc(2*sizeof(char));
        if(strtoken == NULL  ||  buf == NULL) SetError(MBP_EMEM);

        strcpy(buf,"{");
        start = *s;
        for(*s=*s; **s!='\0'; (*s)++)
        {
                i = 0;
                while(isspace(**s)) (*s)++;

                if(IsNameChar(**s))
                {
                        while(IsNameChar(**s))
                        {
                                *(strtoken+i) = **s;
                                (*s)++;
                                i++;
                        }
                        *(strtoken+i) = '\0';

                        if(IsNumber(strtoken))
                        {
                                buf           =           (char           *)realloc((char
*)buf,(strlen(buf)+strlen(strtoken)+1)*sizeof(char));
                                strcat(buf,strtoken);
                                continue;
                        }
```

```
                        /* NAME */
                        sltmp = S;
                        for(S=S; S!=NULL; S=S->next)
                        {
                                if(!strcmp(strtoken,S->sname))
                                {
                                        counter++;
                                        StripBibString(S->svalue,&tmp);
                                        buf         =         (char         *)realloc((char
*)buf,(strlen(buf)+strlen(tmp)+1)*sizeof(char));
                                        strcat(buf,tmp);
                                        free(tmp);
                                        break;
                                }
                        }
                        S = sltmp;
                        continue;
                }



                chunks = 1;

                if(**s == '{')
                {
                        opened = 0;
                        previouschar = '{';
                        (*s)++;

                        while(**s != '}'  ||  previouschar == '\\'  ||  opened != 0)
                        {
                                /*if(**s == '\0')      SetError(MBP_EPARSE);*/
                                if(**s == '{'  &&  previouschar != '\\') opened++;
                                if(**s == '}'  &&  previouschar != '\\') opened--;

                                if(i >= (chunks*STRING_CHUNK) - 1)
                                {
                                        chunks++;
                                        strtoken       =       (char       *)realloc((char
*)strtoken,(chunks*STRING_CHUNK)*sizeof(char));
                                        if(strtoken == NULL) SetError(MBP_EMEM);
                                }

                                previouschar = **s;
                                *(strtoken+i) = **s;
                                i++;
                                (*s)++;
                        }
                        *(strtoken+i) = '\0';
                        (*s)++;


                        buf             =             (char             *)realloc((char
*)buf,(strlen(buf)+strlen(strtoken)+1)*sizeof(char));
                        strcat(buf,strtoken);
                        continue;
                }

                if(**s == '\"')
                {
                        previouschar = '\"';
                        (*s)++;

                        while(**s != '\"'  ||  previouschar == '\\')
                        {
                                /*if(**s == '\0')      SetError(MBP_EPARSE);*/

                                if(i >= (chunks*STRING_CHUNK) - 1)
                                {
                                        chunks++;
                                        strtoken       =       (char       *)realloc((char
*)strtoken,(chunks*STRING_CHUNK)*sizeof(char));
                                        if(strtoken == NULL) SetError(MBP_EMEM);
                                }

                                previouschar = **s;
```

```
                                *(strtoken+i) = **s;
                                i++;
                                (*s)++;
                        }
                        *(strtoken+i) = '\0';
                        (*s)++;

                        buf             =             (char             *)realloc((char
*)buf,(strlen(buf)+strlen(strtoken)+1)*sizeof(char));
                        strcat(buf,strtoken);
                        continue;
                }

        }
        *s = start;
        buf = (char *)realloc((char *)buf,(strlen(buf)+1+1)*sizeof(char));
        strcat(buf,"}");
        free(*s);
        *s = Strdup(buf);
        free(buf);
        free(strtoken);


        return counter;
}



void StripBibString(char *source, char **dest)
{
        int i, l;

        if(*source != '{'  &&  *source != '\"')
        {
                *dest = Strdup(source);
                return;
        }

        l = strlen(source);
        *dest = (char *)malloc((l-2)*sizeof(char));
        if(*dest == NULL) SetError(MBP_EMEM);
        for(i=1; i<l-1; i++)
                *(*dest+i-1) = *(source+i);
        *(*dest+i-1) = '\0';
}


void DeleteStrings(SL_t *S)
{
        if(*S == NULL) return;
        DeleteStrings(&((*S)->next));
        free((*S)->svalue);
        free(*S);
}




/* NO NEED FOR THIS ONE */

/*
void PrintStrings(SL_t S)
{
        if (S==NULL) return;
        printf("%s\t%s\n",S->sname,S->svalue);
        PrintStrings(S->next);
}
*/
```

A.1.16 symbol.h

```
#if defined(MBP_H)
#ifndef SYMBOL_H
#define SYMBOL_H

#define MAXIMUM_FIELDS MAXIMUM_NUMBER_OF_FIELDS_PER_RECORD
#define MAXNAME        MAXIMUM_LENGTH_OF_NAME_OR_NUMBER


/* Entry Types */
#define ARTICLE              1
#define BOOK          2
#define BOOKLET              3
#define CONFERENCE    4
#define INBOOK        5
#define INCOLLECTION  6
#define INPROCEEDINGS 7
#define MANUAL        8
#define MASTERSTHESIS 9
#define MISC          10
#define PHDTHESIS     11
#define PROCEEDINGS   12
#define TECHREPORT    13
#define UNPUBLISHED   14
#define STRING_E      15
#define PREAMBLE      16



/* Field Types */
#define UNKNOWN       -1
#define ADRESS        1
#define ANNOTE        2
#define AUTHOR        3
#define BOOKTITLE     4
#define CHAPTER       5
#define CROSSREF      6
#define EDITION       7
#define EDITOR        8
#define HOWPUBLISHED  9
#define INSTITUTION   10
#define JOURNAL       11
#define KEY           12
#define MONTH         13
#define NOTE          14
#define NUMBER_F      15
#define ORGANIZATION  16
#define PAGES         17
#define PUBLISHER     18
#define SCHOOL        19
#define SERIES        20
#define TITLE         21
#define TYPE          22
#define VOLUME        23
#define YEAR          24



struct field
{
       int    fid;
       char   *value;
       char   fieldname[MAXNAME];
       char   *expandedvalue;
};
typedef struct field   field_t;


struct entry
{
       int            type;
       char           *key;
       field_t        fields[MAXIMUM_FIELDS];
       int            nf;                          /* Num of Fields */
       struct entry   *next;
```

```
};
typedef struct entry   entry_t;
typedef entry_t        *Entry_t;



void InsertField(Entry_t *E, int fid, char *value, char *fieldname);
int AddEntry(Entry_t *E, int type, char *key);
int Name2EntryType(char *name);
int Name2FieldId(char *name);
void DeleteEntries(Entry_t *EL);
/*void PrintEntries(Entry_t EL);*/


#endif
#endif
```

A.1.17 symbol.c

```
#include "mbp.h"


void InsertField(Entry_t *E, int fid, char *value, char *fieldname)
{
        Entry_t temp;


        if(*E == NULL  ||  (*E)->nf >= MAXIMUM_FIELDS)
                SetError(MBP_ESEGM);

        *(value+strlen(value)-1) = '\0';      /* Remove the final char that is '#' */

        temp = *E;
        while((*E)->next!=NULL)
                *E = (*E)->next;

        (*E)->fields[(*E)->nf].fid = fid;
        (*E)->fields[(*E)->nf].value = Strdup(value);
        (*E)->fields[(*E)->nf].expandedvalue = Strdup(value);
        strcpy((*E)->fields[(*E)->nf].fieldname,fieldname);
        (*E)->nf += 1;
        *E = temp;
}


int AddEntry(Entry_t *E, int type, char *key)
{
        Entry_t NE, temp;
        int counter;


        if(*E == NULL)
        {
                *E = (Entry_t)malloc(sizeof(entry_t));
                if(*E == NULL) SetError(MBP_EMEM);
                (*E)->type = type;
                (*E)->nf = 0;
                (*E)->key = Strdup(key);
                (*E)->next = NULL;
                return 1;
        }

        NE = (Entry_t)malloc(sizeof(entry_t));
        if(NE == NULL) SetError(MBP_EMEM);
        NE->type = type;
        NE->nf = 0;
        NE->key = Strdup(key);
        NE->next = NULL;

        counter = 1;
        temp = *E;
        for(*E=*E; (*E)->next!=NULL; *E=(*E)->next)
```

```
                counter++;
        (*E)->next = NE;
        *E = temp;

        return counter+1;
}



int Name2EntryType(char *name)
{
        Lowercase(&name);

        if(strcmp(name,"article") == 0)              return ARTICLE;
        if(strcmp(name,"book") == 0)          return BOOK;
        if(strcmp(name,"booklet") == 0)              return BOOKLET;
        if(strcmp(name,"conference") == 0)    return CONFERENCE;
        if(strcmp(name,"inbook") == 0)               return INBOOK;
        if(strcmp(name,"incollection") == 0)  return INCOLLECTION;
        if(strcmp(name,"inproceedings") == 0) return INPROCEEDINGS;
        if(strcmp(name,"manual") == 0)               return MANUAL;
        if(strcmp(name,"mastersthesis") == 0) return MASTERSTHESIS;
        if(strcmp(name,"misc") == 0)          return MISC;
        if(strcmp(name,"phdthesis") == 0)     return PHDTHESIS;
        if(strcmp(name,"proceedings") == 0)   return PROCEEDINGS;
        if(strcmp(name,"techreport") == 0)    return TECHREPORT;
        if(strcmp(name,"unpublished") == 0)   return UNPUBLISHED;

        if(strcmp(name,"string") == 0)               return STRING_E;
        if(strcmp(name,"preamble") == 0)      return PREAMBLE;

        return UNKNOWN;
}


int Name2FieldId(char *name)
{
        Lowercase(&name);

        if(strcmp(name,"adress") == 0)               return ADRESS;
        if(strcmp(name,"annote") == 0)               return ANNOTE;
        if(strcmp(name,"author") == 0)               return AUTHOR;
        if(strcmp(name,"booktitle") == 0)     return BOOKTITLE;
        if(strcmp(name,"chapter") == 0)              return CHAPTER;
        if(strcmp(name,"crossref") == 0)      return CROSSREF;
        if(strcmp(name,"edition") == 0)              return EDITION;
        if(strcmp(name,"editor") == 0)               return EDITOR;
        if(strcmp(name,"howpublished") == 0)  return HOWPUBLISHED;
        if(strcmp(name,"institution") == 0)   return INSTITUTION;
        if(strcmp(name,"journal") == 0)              return JOURNAL;
        if(strcmp(name,"key") == 0)           return KEY;
        if(strcmp(name,"month") == 0)         return MONTH;
        if(strcmp(name,"note") == 0)          return NOTE;
        if(strcmp(name,"number") == 0)               return NUMBER_F;
        if(strcmp(name,"organization") == 0)  return ORGANIZATION;
        if(strcmp(name,"pages") == 0)         return PAGES;
        if(strcmp(name,"publisher") == 0)     return PUBLISHER;
        if(strcmp(name,"school") == 0)               return SCHOOL;
        if(strcmp(name,"series") == 0)               return SERIES;
        if(strcmp(name,"title") == 0)         return TITLE;
        if(strcmp(name,"type") == 0)          return TYPE;
        if(strcmp(name,"volume") == 0)               return VOLUME;
        if(strcmp(name,"year") == 0)          return YEAR;
        if(strcmp(name,"string") == 0)               return STRING_E;
        if(strcmp(name,"preamble") == 0)      return PREAMBLE;

        return UNKNOWN;
}


void DeleteEntries(Entry_t *EL)
{
        if(*EL == NULL)        return;
        DeleteEntries(&((*EL)->next));
        /* free((*EL)->key); */
        free(*EL);
```

```
}



/* NO NEED FOR THIS ONE */
/*
void PrintEntries(Entry_t EL)
{
        static int counter=1;
        int i;

        if(EL == NULL) return;

        printf("\nEntry %d ---\n",counter++);
        printf("Id:%d\tKey:%s\tFields:%d\n",EL->entry_type,EL->key,EL->num);
        for(i=0; i<EL->num; i++)
        {
                printf("FID:%d\tFName:%s\tValue:%s\tExpanded:%s\n",EL->fields[i].id,EL-
>fields[i].fieldname,EL->fields[i].value,EL->fields[i].expandedvalue);
        }

        PrintEntries(EL->next);
}
*/
```

A.2 Εγκατάσταση

Για την μετάφραση (compile) των παραπάνω αρχείων πηγαίου κώδικα παρέχεται το εξής βοηθητικό αρχείο **Makefile**:

```
PREFIX      = /usr/local
LIBDIR      = $(PREFIX)/lib
INCLDIR     = $(PREFIX)/include/mbp
#VERSION    = 0.0.1

LIB         = libmbp.a
CXX         = gcc
CXXFLAGS    = -Wall -g -c
LIBFLAGS    =
LDFLAGS           =
AR          = ar
ARFLAGS     = rcs


#DIST       = mbp-$(VERSION)
INSTALL     = install
INSTALLOPTS = -m 0755 -o root
INSTALLDIRS = $(INSTALL) $(INSTALLOPTS) -d $(LIBDIR) &&\
              $(INSTALL) $(INSTALLOPTS) -d $(INCLDIR)
SRCS        = common.c lex.c parser.c symbol.c string.c api.c error.c
OBJS        = common.o lex.o parser.o symbol.o string.o api.o error.o
HEADERS     =  global.h  common.h  lex.h  parser.h  symbol.h  string.h
api.h error.h conf.h mbp.h
#TEXT       = README COPYRIGHT COPYING.LIB
```

```
$(LIB):$(OBJS)
	$(AR) $(ARFLAGS) $@ $(OBJS)
	ranlib $@
	@make clean

common.o:common.c $(HEADERS)
	$(CXX) $(CXXFLAGS) $(LDFLAGS) common.c

lex.o:lex.c $(HEADERS)
	$(CXX) $(CXXFLAGS) $(LDFLAGS) lex.c

parser.o:parser.c $(HEADERS)
	$(CXX) $(CXXFLAGS) $(LDFLAGS) parser.c

symbol.o:symbol.c $(HEADERS)
	$(CXX) $(CXXFLAGS) $(LDFLAGS) symbol.c

string.o:string.c $(HEADERS)
	$(CXX) $(CXXFLAGS) $(LDFLAGS) string.c

api.o:api.c $(HEADERS)
	$(CXX) $(CXXFLAGS) $(LDFLAGS) api.c

error.o:error.c $(HEADERS)
	$(CXX) $(CXXFLAGS) $(LDFLAGS) error.c

clean:
	@rm -f $(OBJS)

install:
	@$(INSTALLDIRS)
	@$(INSTALL) $(INSTALLOPTS) $(LIB) $(LIBDIR)
	@$(INSTALL) $(INSTALLOPTS) $(HEADERS) $(INCLDIR)

uninstall:
	@rm -f $(LIBDIR)/$(LIB)
	@for i in $(HEADERS); do\
		rm -f $(INCLDIR)/$$i;\
	done
	@rm -rf $(INCLDIR)
```

Αφού γίνουν οι όποιες τυχόν αλλαγές χρειαστούν στο αρχείο **Makefile** , στον κατάλογο κορυφής του πηγαίου κώδικα εκτελούμε την εντολή **make** (ή κάποιο άλλο εργαλείο για την εκτέλεση των Makefiles). Μετά την εκτέλεση του **Makefile** θα δημιουργηθεί στον ίδιο κατάλογο η βιβλιοθήκη (το αρχείο **libmbp.a**). Ο πιο απλός τρόπος τώρα για την εγκατάσταση αυτής, είναι να γίνουμε διαχειριστές του συτήματος (**root**) με την εντολή **su** και μετά εκτέλεση της εντολής **make install**. Διαφορετικά μπορούμε να βάλουμε τη βιβλιοθήκη σε έναν κατάλογο που υπάρχει στο Library Path (συνήθως η μεταβλητή **LD_LIBRARY_PATH** το καθορίζει αυτό) καθώς και τα αρχεία κεφαλίδες (δηλαδή όλα τα *.h αρχεία) σε έναν κατάλογο που να βρίσκεται στο Include Path.