ISSN 1108-4170

# Τὸ Εὔτυπον

Τεῦχος 5, Ὀκτώβριος 2000

## Σ' ΑΥΤΟ ΤΟ ΤΕΥΧΟΣ:

Ἐπιφυλλίδα ..... iii
Σημειώσεις τοῦ τυπογράφου ..... iv
Απόστολος Συρόπουλος
Εντυπώσεις από το TUG2000 ..... 1
Δημήτριος Ἀ. Φιλίππου
Κώδικες συλλαβισμοῦ γιὰ τὴν στοιχειοθεσία αρχαίων ἑλληνικῶν κειμένων μὲ τὸ TEX καὶ τὸ LATEX ..... 7
Απόστολος Συρόπουλος
Λογοτεχνικός προγραμματισμός: Ή πώς να δημιουργείτε προγράμματα καθώς περιγράφετε το τι θέλετε να κάνει ο Η/Υ ..... 17
A. S. Berdnikov
Package ACCENTBX: some problems with accents in TEX and how to solve them ..... 25
William M. Richter
TEXmerge: a system for generating form-based documents ..... 51
TEXνικές: Σελίδες με φόντο ..... 59
Βιβλίο-Παρουσίαση ..... 61

♣

Μία περιοδικὴ ἔκδοση
τοῦ Δημοκριτείου Πανεπιστημίου Θράκης
σὲ συνεργασία μὲ τὸν Σύλλογο Ἑλλήνων Φίλων τοῦ TEX

**Ἐκδόσεις:** Ἑταιρεία Ἀξιοποίησης και Διαχείρισης Περιουσίας
Δημοκριτείου Πανεπιστημίου Θράκης

**ΞΑΝΘΗ**

# Τὸ Εὔτυπον

ISSN 1108-4170

Εὔτυπον (*Eutupon*) is a publication of the Democritus University of Thrace (Greece) in collaboration with the "Greek TEX's Friends" (*εφτ*) Group. It is published twice a year and it is distributed for free to the members of the *εφτ* Group and to the public in general through Internet. Its articles deal with TEX and, in general, with tools for the electronic typesetting of printed matter.

Editor-in-Chief of Εὔτυπον is Prof. Basil K. Papadopoulos (`papadob@demo.cc.duth.gr`) of the Dept. of Civil Eng., Democritus University of Thrace, Greece. The address of Εὔτυπον is:

Greek TEX's Friends
(c/o A. Syropoulos)
366, 28th October Street
GR-671 00 Xanthi
Greece

URL: `obelix.ee.duth.gr/eft`
E-mail: `eft@ocean1.ee.duth.gr`

The membership fees to the *εφτ* Group are:

Admission:
5,000 GRD (€ 15.00)
6-month subscription:
3,000 GRD (€ 9.00)
Annual subscription:
5,000 GRD (€ 15.00)

(All cheques should be made payable to Apostolos Syropoulos.)

Articles already published in Εὔτυπον may be copied and redistributed for free, provided their origin is mentionned. However, neither Εὔτυπον nor the *εφτ* Group assume any responsibity for methods, products, instructions or ideas described or expressed in articles published herein. The content of articles published herein is the sole responsibility of the respective authors.

Τὸ Εὔτυπον ἀποτελεῖ μία περιοδικὴ ἔκδοση τοῦ Δημοκριτείου Πανεπιστημιου Θράκης σὲ συνεργασία μὲ τὸν Σύλλογο Ἑλλήνων Φίλων τοῦ TEX (*εφτ*). Κυκλοφορεῖ ἀνὰ ἑξάμηνο καὶ διανέμεται δωρεὰν στὰ μέλη τοῦ Συλλόγου καὶ στὸ εὐρύτερο κοινὸ μέσῳ τοῦ Διαδικτύου. Στὶς σελίδες του δημοσιεύονται ἄρθρα σχετικὰ μὲ τὸ TEX καὶ τὴν ἠλεκτρονικὴ στοιχειοθεσία ἐντύπων γενικώτερα.

Ἀρχισυντάκτης καὶ ὑπεύθυνος κατὰ τὸν νόμο γιὰ τὸ Εὔτυπον εἶναι ὁ κ. Βασίλειος Κ. Παπαδόπουλος (`papadob@demo.cc.duth.gr`), καθηγητὴς τοῦ Τμήμ. Πολιτικῶν Μηχανικῶν τοῦ Δ.Π.Θ. Ἡ διεύθυνση τοῦ Εὐτύπου εἶναι:

Σύλλογος Ἑλλήνων Φίλων τοῦ TEX
(ὑπ' ὄψη: Ἀ. Συρόπουλος)
28ης Ὀκτωβρίου 366
671 00 Ξάνθη

URL: `obelix.ee.duth.gr/eft`
E-mail: `eft@ocean1.ee.duth.gr`

Συνδρομὲς μέλους στὸν Σύλλογο *εφτ*:

Ἐγγραφή:
5.000 δρχ. (15 €)
Συνδρομὴ ἑξάμηνη:
3.000 δρχ. (9 €)
Συνδρομὴ ἐτήσια:
5.000 δρχ. (15 €)

(Ἐπιταγὲς θὰ πρέπει νὰ ἀποστέλλονται στὸ ὄνομα τοῦ Ἀποστόλου Συρόπουλου.)

Ἐπιτρέπεται ἡ ἀναδημοσίευση καὶ διανομὴ ἄρθρων ποὺ ἔχουν ἤδη δημοσιευθεῖ στὸ Εὔτυπον, ὑπὸ τὸν ὅρο ὅτι θὰ ἀναφέρεται ἡ προέλευσή τους. Ὡστόσο, τόσο τὸ Εὔτυπον ὅσο καὶ ὁ Σύλλογος *εφτ* δὲν ἀναλαμβάνουν καμία εὐθύνη γιὰ μεθόδους, προϊόντα, ὁδηγίες καὶ ἰδέες ποὺ περιγράφονται ἢ ἐκφράζονται ἐντὸς ἄρθρων δημοσιευμένων στὸ περιοδικό. Τὸ περιεχόμενο τῶν ἄρθρων ποὺ δημοσιεύονται ἐντὸς τοῦ περιοδικοῦ ἀποτελεῖ εὐθύνη τῶν ἀντιστοίχων συγγραφέων.

# ΕΠΙΦΥΛΛΙΔΑ

TEXνίτριες καὶ TEXνίτες!

Φέτος ὁ Σύλλογός μας ἔκλεισε τρία χρόνια ζωῆς καὶ τὸ *Εὔτυπον* ἔκλεισε αἰσίως τὰ δύο. Κουτσὰ-στραβὰ τὸ καραβάκι μας συνεχίζει τὸν αγώνα του καὶ εἴμαστε βέβαιοι ὅτι θὰ βρεῖ θάλασσα ἥρεμη ἐφ' ὅσον οἱ ναῦτες του παλέψουν σκληρά. Ἐὰν ὅμως οἱ ναῦτες ἐγκαταλείψουν τὰ κουπιὰ καὶ τὰ πανιά, τότε τὸ καραβάκι μας θὰ τσακιστεῖ σὲ καμιὰ ξέρα.

Σᾶς ἐντυπωσιάσαμε μὲ τὸν λογοτεχνικό μας οἶστρο; Ὄχι, φίλοι συνTEXνίτες, δὲν εἴμαστε λογοτέχνες! Λογοτεχνίζουμε ἁπλῶς γιὰ νὰ τραβήξουμε τὴν προσοχή σας καὶ νὰ σᾶς ποῦμε ὅτι, ὅπως μπορεῖτε νὰ διαπιστώσετε ἀπὸ τὸ *Εὔτυπον* ποὺ κρατᾶτε στὰ χέρια σας, ὁ Σύλλογος ἔχει μὲν ἀνάγκη ἀπὸ τὸν ὀβολό σας, ἀλλὰ ἔχει ἀνάγκη κατὰ πολὺ περισσότερο ἀπὸ τὰ γραπτά σας. Γι' αὐτό, ἐσεῖς, ποὺ εἶστε τὸ πλήρωμα τοῦ $*\varepsilon\varphi\tau*$, γράψτε μας γιὰ τὶς ὅποιες ἐμπειρίες ἔχετε μὲ τὸ TEX καὶ τὸ LATEX, γιὰ τὶς ἐμπειρίες σας μὲ ἄλλα στοιχειοθετικὰ συστήματα (π.χ., μὲ τὸ Ω ἢ τὴν HTML), γιὰ θέματα ἐπεξεργασίας τοῦ λόγου, γιὰ σχετικὰ προγράμματα ποὺ φτιάξατε ἢ ποὺ θὰ θέλατε νὰ φτιάξετε, γιὰ κάποιο βιβλίο ἢ ἔντυπο ποὺ σᾶς ἐντυπωσίασε μὲ τὴν αἰσθητική του. Νὰ εἶστε σίγουροι, φίλοι συνTEXνίτες, πὼς τὸ *Εὔτυπον* δὲν θὰ ἀρνηθεῖ τὶς σελίδες του σὲ κανέναν.

Ὅμως ἂς δοῦμε τί κρύβει σήμερα τὸ *Εὔτυπον* στὰ ἀμπάρια του. Κατ' ἀρχὰς ἔχει τὸ TUG2000, τὸ ἐτήσιο συνέδριο τοῦ TEX Users Group ποὺ φέτος ἔγινε στὴν Ὀξφόρδη. Ὁ Ἀπόστολος Συρόπουλος, ποὺ παρακολούθησε τὸ συνέδριο, μᾶς τὸ περιγράφει καὶ προτείνει τὸ μεθεπόμενο συνέδριο τοῦ TUG νὰ γίνει στὴν Ξάνθη — ἐγχείρημα δύσκολο ἀλλὰ ὄχι ἀκατόρθωτο. Ὁ Δημήτριος Φιλίππου ξαναχτυπᾶ στὸ δόξα-Πατρὶ τοῦ συλλαβισμοῦ τῶν ἀρχαίων ἑλληνικῶν (συγχωρέστε τον, παιδὶ εἶναι κι αὐτός). Ὁ Ἀπόστολος Συρόπουλος ἐπανέρχεται μὲ τὸν *λογοτεχνικὸ προγραμματισμό* (σᾶς ἀρέσει ὁ ὅρος ἢ νὰ ψάξουμε γιὰ κανένα ἄλλο καλύτερο;). Ὁ Alexander (Sasha) Berdnikov, παλιὸς ρῶσος ἀρχιTEXνίτης, μᾶς παρουσιάζει μία λύση στὸ πρόβλημα τῆς τοποθέτησης πολλαπλῶν διακριτικῶν σημείων σὲ διάφορα γράμματα, ἐνῶ ὁ William Richter μᾶς παρουσιάζει ἕνα πρόγραμμα γιὰ τὴν ἐνσωμάτωση δεδομένων σὲ ἕνα ἤδη ὑπάρχον κείμενο TEX. Καὶ ὡς συνήθως, τὸ τεῦχος κλείνει μὲ τὰ τακτικὰ καὶ ἄκρως ἐνδιαφέροντα ἄρθρα τῶν TEXχνικῶν καὶ τῆς Βιβλίο-Παρουσίασης.

Αὐτὰ λοιπὸν γιὰ σήμερα. Ἕως τὴν ἐπόμενη ἐπικοινωνία μας, σᾶς εὐχόμαστε ὅ,τι καλύτερο καὶ TEXνικώτερο!

– Ἀπόστολος Συρόπουλος
– Δημήτριος Ἀ. Φιλίππου

## Οἱ σημειώσεις τοῦ τυπογράφου

Τὸ Εὔτυπον στοιχειοθετήθηκε μὲ τὸ LaTeX. Ὡς βασικὰ πακέτα χρησιμοποιήθηκαν τὸ `cah-gut.cls` (τὸ πακέτο τοῦ γαλλικοῦ περιοδικοῦ *Les Cahiers GUTenberg*) καὶ τὸ babel μὲ τὴν ἀγγλικὴ καὶ τὴν ἑλληνικὴ ἐπιλογή. Γιὰ τὶς διάφορες γραφικὲς παραστάσεις, χρησιμοποιήθηκαν ἐπίσης τὰ πακέτα `graphicx` καὶ `color`. Ὁ Ὀβελὶξ στὴν σελίδα 59 εἶναι τοῦ Uderzo, ἐνῶ τὸ σκίτσο στὴν τελευταία σελίδα εἶναι τοῦ Στάθη Σταυρόπουλου καὶ προέρχεται ἀπὸ τὴν ἐφημερίδα *Τὰ Νέα* (27/9/1999, σελ. P3). Ὁ γραμμικὸς κώδικας τοῦ ὀπισθόφυλλου εἶναι μιὰ εὐγενικὴ προσφορὰ τοῦ Γιάννη Χαραλάμπους.

Ἡ ἐκτύπωση ἔγινε μὲ τὸ πρόγραμμα Ghostscript σὲ ἐκτυπωτὴ HP LaserJet V (1200 dpi), ἐνῶ ἡ βιβλιοδεσία ἔγινε στὸ Τυπογραφεῖο τοῦ Δημοκριτείου Πανεπιστημίου Θράκης στὴν Ξάνθη.

# Εντυπώσεις από το TUG2000

Απόστολος Συρόπουλος

*28ης Οκτωβρίου 366*
*671 00 ΞΑΝΘΗ*
*apostolo@obelix.ee.duth.gr*

**Abstract**

The annual TEX Users Group meeting is the most importand event in the world of digital typography with TEX. The author, thanks to a generous bursary from the TUG Bursary Committee, had the chance to attend this year meeting which took place in Oxford, U.K., from the 12th of August until the 16th of August 2000. This text is a report of all major events that took place as well as a brief description of the most importand talks—at least in the author's view.

## 1. Γενικά

Όπως κάθε χρόνο έτσι και φέτος έλαβε χώρα το ετήσιο συνέδριο του TEX Users Group στην Οξφόρδη του Ενωμένου Βασιλείου από τις 12 ως τις 16 Αυγούστου. Ο συγγραφέας του παρόντος είχε την ευκαιρία να παρακολουθήσει τις εργασίες του συνεδρίου, αλλά και να παρουσιάσει μια εργασία που σχετίζεται με ένα πρόγραμμα το οποίο επιτρέπει την εύκολη δημιουργία γραφικών μέσω του πακέτου μακροεντολών PICTEX. Στο σημείο αυτό θα πρέπει να τονιστεί ότι ο συγγραφέας είχε την τύχη να παρακολουθήσει το συνέδριο επειδή έτυχε οικονομικής ενισχύσεως, από την επιτροπή υποτροφιών του TUG, η οποία κάλυψε τα έξοδα διαμονής, μεταφοράς και εγγραφής στο συνέδριο. Για τον λόγο αυτό θέλω να ευχαριστήσω την επιτροπή υποτροφιών του TUG και ειδικά τον Phil Taylor για τις χρήσιμες συμβουλές του.

## 2. Το συνέδριο

Το συνέδριο ξεκίνησε με μια ομιλία της Mimi Jett, προέδρου του TUG, στην οποία καλωσόρισε τους σύνεδρους και ευχήθηκε κάθε επιτυχία στο συνέδριο. Στη

**Πανοραμική άποψη της πόλης της Οξφόρδης.**

συνέχεια, η Barbara Beeton, που είναι γνωστή από τους αγώνες της για την εξάπλωση του TEX και ως παλιά συνεργάτιδα του Knuth, αναφέρθηκε στην ανάγκη ενσωμάτωσης μαθηματικών συμβόλων στο Unicode καθώς και τα προβλήματα που ανακύπτουν από τις ιδιορυθμίες του κάθε συγγραφέα. Στην συνέχεια ο Jonathan Fine μίλησε για τα προβλήματα που ανακύπτουν από την αλληλεπίδραση των διαδικασιών *σπασίματος* των γραμμών και σελίδων. Η μέρα τελείωσε με το... συνηθισμένο αγγλικό τσάι.

Τις επόμενες μέρες παρουσιάστηκαν αρκετά ενδιαφέροντα πράγματα. Από αυτά αξίζει κανείς να αναφέρει τα εξής: την ομιλία του John Plaice για το Ω2, τις ομιλίες σχετικά με το pdfTEX, την παρουσίαση του $\mathcal{NTS}$ και την παρουσίαση του PassiveTEX. Επειδή βέβαια οι ομιλίες αυτές παρουσίασαν το μεγαλύτερο ενδιαφέρον θα τις παρουσιάσουμε σε κάποια σχετική έκταση.

Η ομιλία του John Plaice ήταν πολύ διαφωτιστική όσον αφορά την τρέχουσα έκδοση του Ω. Είναι βέβαια περιττό να πούμε ότι το Ω είναι ένα σύστημα το οποίο ενδιαφέρει κύρια τους μη-αγγλόφωνους TEXνίτες. Από αυτά που είπε ο John ξεχώρισα τη δυνατότητα να καλεί κανείς εξωτερικά προγράμματα σε Perl τα οποία δίνουν επιπρόσθετη λειτουργικότητα στο σύστημα. Βέβαια κατά την προσωπική μου άποψη αυτό αποτελεί και ένα μειονέκτημα του συστήματος, αφού έτσι τα αρχεία μας μπορούν να γίνουν λιγότερο portable. Δυστυχώς όμως αυτό που είναι λυπηρό είναι το γεγονός ότι το σύστημα δεν έχει ακόμη την κατάλληλη

τεκμηρίωση και έτσι πολλά πράγματα μοιάζουν σχεδόν αποκρυφιστικά. Εύχομαι, πολύ σύντομα να αλλάξει αυτή η κατάσταση ώστε το Ω να πάρει την θέση που του αξίζει.

Για το pdfTEX έχει γράψει αρκετά πράγματα ο φίλος Γιάννης Δημάκος και έτσι πιστεύω οι περισσότεροι αναγνώστες είναι εξοικιωμένοι με την λειτουργικότητα του συστήματος. Ο Hans Hagen ήταν ο κύριος ομιλητής, αν και ο Hàn Thế Thành είναι αυτός που αρχικά δημιούργησε το σύστημα. Φυσικά μίλησε και αυτός, αλλά αυτό δεν αλλάζει τίποτα.

Ο Phil Taylor και Karel Skoupý μιλήσαν για το $\mathcal{NTS}$ και την ιστορία του. Για τους μη γνωρίζοντες αναφέρω ότι το $\mathcal{NTS}$ είναι μια προσπάθεια να ξαναγραφτεί το TEX σε μία σύγχρονη γλώσσα προγραματισμού ώστε να αυτό να πάψει να είναι μονολιθικό. Το σύστημα υλοποιήθηκε στην γλώσσα προγραμματισμού Java, αλλά προς το παρόν είναι εξαιρετικά αργό. Φυσικά με την προοπτική της παραπέρα αύξησης της ταχύτητας της γλώσσας αλλά και του γεγονότος ότι σύντομα θα υπάρχουν μηχανές Java, αυτό δεν φαίνεται να είναι μεγάλο πρόβλημα.

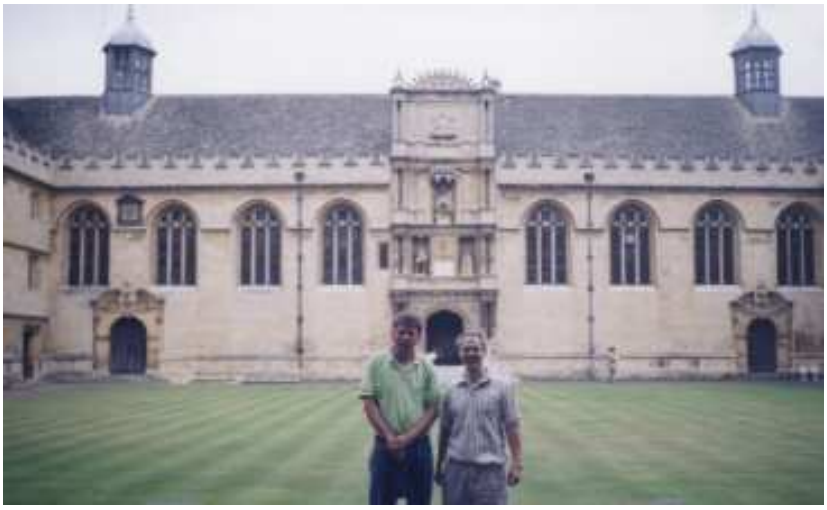
Ακούγοντας όλες αυτές τις ομιλίες κανείς αποκομίζει αμέσως την εντύπωση ότι αυτή τη στιγμή υπάρχουν τρεις ομάδες που εργάζονται σε εντελώς διαφορετικά προγράμματα (ελληνιστί: projects). Αυτό πραγματικά είναι μια εντελώς λυπηρή διαπίστωση. Μιας και η Java κατανοεί το Unicode, θα μπορούσε κανείς να πεί ότι είναι δυνατό τα προγράμματα για το $\mathcal{NTS}$ και το Ω να γίνουν ένα ενιαίο πρόγραμμα. Επιπλέον, επειδή είναι φανερή πλέον η δύναμη του PDF, θα μπορούσε κάλλιστα να ενσωματωθεί η δυνατότητα παραγωγής αρχείων PDF από το πρώτο εννιαίο πρόγραμμα, ώστε τελικά να υπάρχει μόνο ένα πρόγραμμα στο οποίο θα δουλέψουμε όλοι μας ώστε να βγει τελικά ένα πραγματικά άψογο open source σύστημα στοιχειοθεσίας για τον αιώνα που έρχεται. Αλλά φαίνεται ότι ο καθένας θέλει τη δόξα για τον εαυτό του...

Ο Sebastian Rahtz είναι ένας εξαιρετικός TEXνίτης που τελευταία έχει πάθος με την γλώσσα επισήμανσης XML. Για τον λόγο αυτό, σε συνεργασία με τον γνωστό μας Michel Goossens, προχώρησαν στη δημιουργία του πακέτου PassiveTEX. Το σύστημα αυτό ουσιαστικά αντικαθιστά το jadetex[1] του Sebastian Rahtz και επιτρέπει την συνεργασία του TEX με την XML.

Μερικές ακόμη αξιόλογες ομιλίες αφορούσαν το AsTEX, ενός *διαμέσου* (interface) για λειτουργικά συστήματα της Micro$oft, τον τρόπο δημιουργίας ενός μεταγλωτιστή της Pascal γραμμένο εξ' ολοκλήρου σε TEX και τέλος τη δημιουργία μιας γραμματοσειράς και ενός πακέτου για τη στοιχειοθεσία προβλημάτων σκάκι.

Εκτός βέβαια από τις διαλέξεις είχα την ευκαιρία να συνατηθώ και να συνομιλήσω με μερικούς TEXνίτες των οποίων η δουλειά σχετίζεται άμεσα ή έμμεσα με

[1] Βλέπε: Α. Συρόπουλος, *Τὸ Εὔτυπον*, νο. 4, σελ. 63–70.



**Ο συγγραφέας του άρθρου με τον συνεργάτη του Dick Nickalls στην αυλή του Wadham College.**

πράγματα που μας ενδιαφέρουν. Έτσι αντάλλαξα κάποιες απόψεις με τον Johannes Braams, τον άνθρωπο που έφτιαξε το πακέτο babel, τον Nelson Beebe, τον άνθρωπο που κρατάει ένα αρχείο με όλες σχεδόν τις βιβλιογραφικές αναφορές που σχετίζονται με την επιστήμη των ηλεκτρονικών υπολογιστών, την Mimi Jett και φυσικά με τον John Plaice.

Εκτός όμως από διαλέξεις το συνέδριο είχε και ορισμένα... happenings. Μια μέρα πριν από το τέλος του συνεδρίου, βρεθήκαμε στο Μουσείο Φυσικής Ιστορίας της Οξφόρδης, όπου μεταξύ σκελετών δυνοσαύρων, συζητούσαμε για το κοινό μας πάθος. Ελπίζω η συγκέντρωση αυτή στο συγκεκριμένο μέρος να μην ήταν... συμπτωματική! Αμέσως μετά ακολούθησε το επίσημο δείπνο, στο hall του Wadham College, στο χώρο δηλαδή που γινόταν το συνέδριο. Εκεί κλήθηκαν να μιλήσουν ένας σύνεδρος ανά γεωγραφική ήπειρο. Αυτός που έδωσε ιδιαίτερο τόνο στη βραδυά ήταν ο εκπρόσωπος της Ασίας, ένας Ιρανός που κατοικεί στην Αγγλία εδώ και τριάντα χρόνια. Παραπονέθηκε ότι εδώ και πολλά χρόνια προσπαθεί να αποκτήσει πραγματικά αγγλική προφορά και τώρα είναι ο εκπρόσωπος της Ασίας! Όμως το ανέκδοτο που μας είπε ήταν απλά καταπληκτικό! Το ίδιο περίπου έγινε και με τον John Plaice ο οποίος σύμφωνα με τα λόγια του Phil Taylor μιλάει αγγλικά με γαλλική προφορά και γαλλικά με αγγλική προφορά (για τους μη γνωρίζοντες, σημειώνουμε ότι ο John είναι Καναδός). Ο John λοιπόν εκπροσώπησε την Ωκεανία, λόγω του ότι απλά εργάζεται στην Αυστραλία. Δύο

ακόμη ατάκες προκάλεσαν το γέλιο του κόσμου. Η πρώτη αφορά τον Phil Taylor, τον οικοδεσπότη μας. Έτσι, σύμφωνα με τα λεγόμενά του, το ότι επιλέχθηκε για οικοδεσπότης δεν οφείλονταν στο ότι κατέχει μια ιδιαίτερη θέση στον χώρο ή επειδή παντρεύτηκε πρόσφατα, αλλά απλά επειδή έχει τη... δυνατότερη φωνή! Σε κάποιο σημείο ο Sebastian Rahtz τιμήθηκε για την προσφορά του και ο Phil Taylor αναφερόμενος στην οικογένεια του Sebastian είπε ότι του αρέσει να τους πηγαίνει στα νεκροταφεία για εκδρομή! Εκείνος, στην αντιφώνησή του, είπε, μεταξύ άλλων, ότι αυτό δεν το θέλει αυτός, αλλά απλά αρέσει στα παιδιά του να παίζουν στα νεκροταφεία![2]

Ως γενική κριτική μπορώ να πω ότι το συνέδριο ήταν ένα πραγματικά επιτυχημένο γεγονός. Αυτό άλλωστε το μαρτυρά και η ποιότητα των πρακτικών, τα οποία θα εμφανιστούν σε προσεχές τεύχος του *TUGboat*. Με αυτή την αφορμή σας προτείνω ανεπιφύλακτα αν σας δωθεί η ευκαιρία να παρακολουθήσετε τις εργασίες του επόμενου συνεδρίου το οποίο θα πραγματοποιηθεί από τις 12 ως τις 15 Αυγούστου του 2001 στο Newark της πολιτείας του Delaware των Η.Π.Α. Ίσως θα ήταν μια καλή ευκαιρία να σκεφτούμε μήπως μπορούμε εμείς να διοργανώσουμε το συνέδριο του 2002 εδώ στην πόλη της Ξάνθης.

[2] Συνήθως δεν πιστεύω στις συμπτώσεις, αλλά την ημέρα που έγραφα τις παραπάνω γραμμές (2 Σεπτεμβρίου 2000), ο αδερφός μου είχε ένα τρομερό ατύχημα που τελικά του στέρησε την ζωή και τον οδήγησε στα... νεκροταφεία.

# Κώδικες συλλαβισμοῦ γιὰ τὴν στοιχειοθεσία ἀρχαίων ἑλληνικῶν κειμένων μὲ τὸ TEX καὶ τὸ LATEX

Δημήτριος Ἀ. Φιλίππου

*Κάτω Γατζέα*
*385 00 Βόλος*

*Καλοί μου φιλόλογοι,*
*βαρᾶτε με, ἀλλὰ λίγο ἀλαφρύτερα*
*ποὺ μπῆκα γι' ἄλλη μιὰ φορὰ*
*στ' ἀμπελοχώραφά σας!*
Ἐπιρρεπὴς καὶ ἐθισμένος

**Abstract**

The grammatical rules for hyphenation of ancient Greek differ considerably from those for modern Greek, particularly when it comes to compound words. Moreover, there are not any freely available hyphenation patterns for typesetting ancient Greek texts with TEX. Thus, new hyphenation patterns had to be written from scratch for typesetting ancient Greek. These new patterns allow for the separation of compound words in their original parts when the first part has not lost its final vowel by ellision. The new hyphenation patterns, which can be used with the `greek` option of `babel` or Dryllerakis' GREEKTEX, are available for free from CTAN.

## 1. Εἰσαγωγὴ

Σὲ προηγούμενο ἄρθρο [1], εἴχαμε ἀναφερθεῖ στὴν δημιουργία βελτιωμένων κωδίκων συλλαβισμοῦ γιὰ τὴν στοιχειοθεσία νεοελληνικῶν κειμένων μὲ τὸ TEX καὶ τὸ LATEX. Οἱ κώδικες συλλαβισμοῦ πολυτονικῶν νεοελληνικῶν κειμένων θὰ μποροῦσαν νὰ χρησιμεύσουν καὶ γιὰ τὸν συλλαβισμὸ ἀρχαίων ἑλληνικῶν κειμένων, ἐφ' ὅσον καὶ οἱ ἀρχαῖοι Ἕλληνες συλλάβιζαν μὲ βάση τὸν κανόνα ποὺ λέει

ὅτι ἕνα σύμφωνο συλλαβίζεται πάντα μὲ τὸ ἀκόλουθο φωνῆεν [2]. Ἴσως δὲ οἱ ἀρχαῖοι Ἕλληνες νὰ χώριζαν τὶς συλλαβὲς μὲ βάση μόνο τὸν τονισμὸ καὶ τὴν εὐφωνία. Γιὰ παράδειγμα, ὁ Smyth [2] ἀναφέρει ὅτι οἱ ἀρχαῖοι Ἕλληνες συλλάβιζαν τὴν ἔκφραση *ἐκ τούτου* σὰν νὰ ἦταν μία λέξη, δηλαδὴ συλλάβιζαν *ἐ-κ‿τού-του*.

Ὡστόσο, νεώτεροι γραμματικολόγοι εἰσήγαγαν ἕναν κανόνα γιὰ τὸν συλλαβισμὸ τῶν σύνθετων ἀρχαίων ἑλληνικῶν λέξεων μὲ βάση τὸν χωρισμὸ τῶν συνθετικῶν τους [3]. Σύμφωνα μ' αὐτὸν τὸν κανόνα, συλλαβίζουμε *ἀπ-αγγέλω*, *Διόσκουρος* καὶ *παρ-εκ-τρέπω*, ἀντὶ γιὰ *ἀ-παγγέλω*, *Διό-σκουρος* καὶ *πα-ρε-κτρέπω*.

Ἡ Ἀκαδημία Ἀθηνῶν [4], ἀλλὰ καὶ ὁρισμένες ἔγκυρες γραμματικὲς τοῦ περασμένου αἰώνα [3], ἀποδέχονται τὸν παραπάνω κανόνα ἐλαφρῶς τροποποιημένο· οἱ σύνθετες λέξεις χωρίζονται στὰ συνθετικά τους μέρη, μόνο ὅταν δὲν ἔχει συμβεῖ ἔκθλιψη φωνηέντων τοῦ πρώτου συνθετικοῦ. Ἡ σχολικὴ γραμματικὴ τῶν ἀρχαίων ἑλληνικῶν [5], ποὺ ἀκολουθεῖ τὶς ὁδηγίες τῆς Ἀκαδημίας Ἀθηνῶν, γράφει χαρακτηριστικά:

> «Οι σύνθετες λέξεις χωρίζονται στα συνθετικά τους μέρη, αν κατά τη σύνθεση δεν έχει αποβληθεί το τελικό φωνήεν του θέματος του α' συνθετικού· αλλιώς, συλλαβίζονται σαν απλές λέξεις: *ἐξ-έρχομαι*, *συν-άγω*, *δυσ-τυχής*, *Ἑλλήσ-ποντος*, *νουν-εχής*· αλλά: *πα-ρέρχομαι*, *ἀ-πέχω*, *κά-θοδος*, *νο-μάρχης*, *φί-λιππος*, *πρω-ταγωνιστής*.»

Ὁ Χαραλάμπους [6] ἔχει ἑτοιμάσει κώδικες συλλαβισμοῦ ἀρχαιοελληνικῶν κειμένων μὲ βάση τὸν χωρισμὸ τῶν σύνθετων ἀρχαιοελληνικῶν λέξεων στὰ συνθετικά τους μέρη ἀκόμα καὶ ὅταν ἔχει ἀποβληθεῖ τὸ τελικὸ φωνήεων τοῦ πρώτου συνθετικοῦ. Δηλαδὴ οἱ κώδικες τοῦ Χαραλάμπους δίνουν: *αἰχμ-άλωτος* (*αἰχμή* + *ἁλωτός*), *στρατ-ηγός* (*στρατὸς* + *ἄγω*), ἐνὼ κατὰ τὴν σχολικὴ γραμματικὴ τῶν ἀρχαίων [5] ἔχουμε: *αἰχ-μάλωτος*, *στρα-τηγός*. Ἐπιπλέον, οἱ κώδικες συλλαβισμοῦ τοῦ Χαραλάμπους γιὰ τὰ ἀρχαῖα ἑλληνικὰ δὲν κυκλοφοροῦν δωρεὰν στὸ Διαδίκτυο.

Προκειμένου ὅλοι οἱ χρῆστες τοῦ TEX καὶ τοῦ LATEX νὰ μποροῦν νὰ στοιχειοθετοῦν ἀρχαῖα ἑλληνικὰ κείμενα μὲ σωστὸ συλλαβισμό, δημιουργήσαμε νέους κώδικες συλλαβισμοῦ. Αὐτοὶ οἱ νέοι κώδικες συλλαβισμοῦ γιὰ τὰ ἀρχαῖα ἑλληνικὰ διαφέρουν ἀπὸ ἐκείνους τοῦ Χαραλάμπους [6] στὸ ὅτι στηρίζονται στοὺς κανόνες τῆς Ἀκαδημίας Ἀθηνῶν καὶ τῆς σχολικῆς γραμματικῆς. Μὲ ἄλλα λόγια οἱ νέοι κώδικες συλλαβισμοῦ δίνουν *προσ-έχω* ὅμως *πα-ρέ-χω*, ἐνὼ οἱ κώδικες τοῦ Χαραλάμπους δίνουν *προσ-έχω* καὶ *παρ-έχω*. Παρακάτω ἐξηγεῖται λεπτομερῶς τὸ πῶς φτιάχθηκαν οἱ νέοι κώδικες συλλαβισμοῦ γιὰ τὰ ἀρχαῖα ἑλληνικά, ἀλλὰ καὶ οἱ σπάνιες περιπτώσεις ὅπου τὸ TEX μπορεῖ νὰ συλλαβίσει λανθασμένα.

## 2. Ἡ δημιουργία βελτιωμένων κωδίκων συλλαβισμοῦ γιὰ τὰ ἀρχαῖα ἑλληνικά

Τὸ πῶς συλλαβίζει τὸ TEX τὸ εἴδαμε στὸ προηγούμενο σχετικὸ ἄρθρο [1]. Ἔτσι, ὅπως στὴν περίπτωση τῶν νέων ἑλληνικῶν, οἱ κώδικες συλλαβισμοῦ τῶν ἀρχαίων ἑλληνικῶν προκύπτουν κατὰ ἀντιστοιχία μὲ τοὺς γραμματικοὺς κανόνες συλλαβισμοῦ:

(α) *Ἕνα σύμφωνο ἀνάμεσα σὲ δύο φωνήεντα συλλαβίζεται μὲ τὸ δεύτερο φωνῆεν*

Μὲ ἄλλα λόγια, μία ἁπλὴ λέξη χωρίζεται σὲ συλλαβὲς μετὰ ἀπὸ κάθε φωνῆεν: π.χ., *ἀ-γνο-έ-ω*. Οἱ ἀντίστοιχοι κώδικες εἶναι: `a1 e1 h1 i1 o1 u1 w1`. Γιὰ τὴν ἀποφυγὴ χωρισμοῦ μεταξὺ φωνήεντος καὶ ὑποδιαστολῆς, προσθέτουμε: `a2|1 h2|1 w2|1`.

(β) *Οἱ δίφθογγοι δὲν χωρίζονται*

Οἱ ἀντίστοιχοι κώδικες εἶναι: `a2i a2'i ... u2<‘i u2<~i`. Μεταξὺ τῶν κωδίκων αὐτῶν παρεμβάλλουμε ὁρισμένες ἐξαιρέσεις: `'a3i >a3i <a3i ... 'u3i ~u3i >u3i <u3i ou3i 'ou4i >ou4i <ou4i o3u'i o3u~i`.

(γ) *Τὰ σύμφωνα — ἁπλὰ καὶ δίψηφα — στὴν ἀρχὴ ἢ στὸ τέλος λέξεων δὲν ἀποτελοῦν συλλαβές*

Οἱ ἀντίστοιχοι κώδικες εἶναι: `4b. 4g. ... 4y. .b4 .g4 ... .y4`. Στοὺς κώδικες αὐτούς, προσθέτουμε μερικοὺς ἀκόμα γιὰ τὴν περίπτωση ἐλλειπῶν λέξεων ποὺ τελειώνουν μὲ ἀπόστροφο ἐξ αἰτίας ἔκθλιψης φωνήεντος: `4'' 4b'' 4g'' ... 4y''`. Ἐπίσης προσθέτουμε καὶ μερικοὺς κώδικες γιὰ τὴν περίπτωση ἐλλειπῶν λέξεων ποὺ τελειώνουν μὲ δύο σύμφωνα καὶ ἀπόστροφο: `4bd'' 4bl'' ... 4qn'' 4qr''`.

Οἱ λέξεις ποὺ ἔχουν ὑποστεῖ ἔκθλιψη ἐμφανίζονται κυρίως στὴν ἀρχαία ἑλληνικὴ ποίηση. Γιὰ παράδειγμα, ὁ ἀριστοφανικὸς ἥρωας Εὐελπίδης λέει:

> ἀστοὶ μετ' ἀστῶν, οὐ σοβοῦντος οὐδενὸς
> ἀνεπτόμεσθ' ἐκ τῆς πατρίδος ἀμφοῖν ποδοῖν,
> αὐτὴν μὲν οὐ μισοῦντ' ἐκείνην τὴν πόλιν...
>
> Ἀριστοφάνης, *Ὄρνιθες*, στίχοι 34–36

Βεβαίως, τὸ κόψιμο τῶν λέξεων εἶναι σπάνιο στὴν ποίηση. Ἐπειδὴ ὅμως τέτοιες ποιητικὲς φράσεις μπορεῖ νὰ παρουσιασθοῦν ὄχι μόνο στὴν μορφὴ στίχων, ἀλλὰ καὶ ἐντὸς πεζοῦ κειμένου, γι' αὐτὸ καὶ οἱ παραπάνω κώδικες εἶναι ἀπαραίτητοι.

(δ) *Τὰ διπλὰ σύμφωνα χωρίζονται*

Οἱ ἀντίστοιχοι κώδικες εἶναι: `4b1b 4g1g ... 4r1r 4>r1<r 4s1s ... 4y1y`. (Ὁ Χαραλάμπους [6] λέει ὅτι τὸ διπλὸ *ρ* — *ρρ* ἢ *ῥῥ* — δὲν χωρίζεται, ἀλλὰ αὐτὸ εἶναι μάλλον λάθος.)

(ε) *Συνδυασμοὶ δύο ἢ περισσοτέρων συμφώνων, ἐκ τῶν ὁποίων δύο πρώτων δὲν ἀρχίζει ἑλληνικὴ λέξη, χωρίζονται μετὰ τὸ πρῶτο σύμφωνο*

Οἱ ἀντίστοιχοι κώδικες εἶναι: `2b1g ... 2y1q`. Θὰ πρέπει νὰ σημειώσουμε ὅτι ὑπάρχουν ὁρισμένοι συνδυασμοὶ συμφώνων ἀπὸ τοὺς ὁποίους ἀρχίζουν λέξεις τῆς νέας ἑλληνικῆς γλώσσας, ἀλλὰ ἀπὸ τοὺς ὁποίους δὲν ἀρχίζουν λέξεις τῆς ἀρχαίας ἑλληνικῆς. Ἔτσι, ὁ συνδυασμὸς *βγ* δὲν χωρίζεται στὰ νέα ἑλληνικά (γιατὶ ὑπάρχει ἡ λέξη *βγαίνω*), ἐνὼ χωρίζεται στὰ ἀρχαῖα ἑλληνικά (γιατὶ δὲν ὑπάρχει ἀρχαία ἑλληνικὴ λέξη ποὺ νὰ ἀρχίζει ἀπὸ *βγ*). Τὰ συμφωνικὰ συμπλέγματα ἀπὸ τὰ ὁποῖα ἀρχίζουν λέξεις ἀρχαιοελληνικές, τὰ ἐπιβεβαιώσαμε ἀπὸ τὸ λεξικὸ Liddell–Scott [7] καὶ τὸ λεξικὸ τοῦ Σταματάκου [8].

Ἕνας ἄλλος κανόνας γιὰ τὸν συλλαβισμὸ τῶν ἀρχαίων λέει [3] ὅτι δὲν χωρίζεται ὁ συνδυασμὸς δύο συμφώνων ὅταν τὸ πρῶτο εἶναι ἄφωνο (*π*, *β*, *φ* / *τ*, *δ*, *θ* / *κ*, *γ*, *χ*) καὶ τὸ δεύτερο εἶναι ἕνα ἀπὸ τὰ ρινικὰ *μ* ἢ *ν*. Θὰ πρέπει λοιπὸν οἱ κώδικες νὰ μὴν περιέχουν τὸν χωρισμὸ τῶν *βμ*, *βν*, *γμ*, κ.τ.ὅ. Γι' αὐτὸ καὶ δὲν συμπεριλαμβάνουμε τοὺς κώδικες `2b1m ... 2q1n`.

(ϛ) *Μία σύνθετη λέξη χωρίζεται στὰ συνθετικά της μέρη. Ὅταν ὅμως ἔχει ἐπέλθει ἔκθλιση φωνηέντων κατὰ τὴν σύνθεση, τότε ἡ λέξη συλλαβίζεται σὰν μία ἁπλὴ λέξη.*

Αὐτὸς ὁ κανόνας δημιουργεῖ τὶς μεγαλύτερες δυσκολίες γιὰ τὴν δημιουργία τῶν κωδίκων συλλαβισμοῦ τῶν ἀρχαίων ἑλληνικῶν. Τὴν λύση στὸ πρόβλημα τὴν ἐπεσήμανε πρῶτος ὁ Χαραλάμπους [6]: Μὲ βάση ἕνα λεξικὸ τῆς ἀρχαίας ἑλληνικῆς γλώσσας — στὴν περίπτωσή μας, τὸ μεγάλο λεξικὸ τοῦ Σταματάκου [8] — καταστρώσαμε ἕναν ἐκτεταμένο κατάλογο ἀπὸ κώδικες ποὺ ἀντιστοιχοῦν στὰ ἀρχικὰ μέρη τέτοιων σύνθετων λέξεων:

| | |
|---|---|
| `>adi'e2x1` | ἀδιέξ-οδος |
| `>adie2x1` | |
| `>adu2s1'w` | ἀδυσ-ώπητος |
| `>adu2s1w` | |
| `<al'o2s1` | ἁλόσ-ανθος |
| `<alo2s1` | |
| `...` | ... |

```
<w2s1e‘i.     ὡσ-εἰ
<w2s1e’i.
<’w2s1per.    ὥσ-περ
<w2s1pe2r1    ὡσ-περ-εἰ
<’w2s1te      ὥσ-τε
```

Ὁ κατάλογος ἀρχικῶν μερῶν σύνθετων λέξεων περιλαμβάνει καὶ μερικὲς νεώτερες λέξεις, ὅπως *καλωσ-ορίζω*, *καν-εἰς* καὶ *ὁλον-έν*. Προσθέσαμε τὶς λέξεις αὐτὲς γιὰ τὴν περίπτωση ποὺ οἱ κώδικες τῶν ἀρχαίων χρησιμοποιηθοῦν γιὰ τὸν συλλαβισμὸ κειμένων τῆς καθαρεύουσας. Ἂς σημειωθεῖ ὅτι ὁ πλήρης κατάλογος ἀρχικῶν μερῶν σύνθετων λέξεων τῆς ἀρχαίας ἑλληνικῆς γλώσσας εἶναι πολὺ μεγάλος καὶ γι’ αὐτὸ δὲν δίνεται ἐδῶ. Ὅσοι ἐνδιαφέρονται μποροῦν νὰ κοιτάξουν τὸ σχετικὸ ἀρχεῖο ποὺ συνοδεύει τὸ ἀρχεῖο μὲ τοὺς κώδικες συλλαβισμοῦ (βλ. §5).

## 3. Δοκιμὴ τῶν κωδίκων συλλαβισμοῦ

Ὅπως μὲ τοὺς κώδικες συλλαβισμοῦ γιὰ τὴν στοιχειοθεσία νεοελληνικῶν κειμένων [1], ἔτσι καὶ μὲ τοὺς κώδικες συλλαβισμοῦ γιὰ τὰ ἀρχαῖα ἑλληνικὰ δημιουργήσαμε μία νέα φόρμα `latex`. Κατόπιν, μὲ τὴν βοήθεια τῆς ἐντολῆς `\showhyphens`, δοκιμάσαμε τοὺς κώδικες συλλαβισμοῦ ἀρχαίων ἑλληνικῶν στὸν συλλαβισμὸ πέντε διαφορετικῶν κειμένων ποὺ ἡ συνολικὴ ἔκτασή τους ξεπερνᾶ τὶς ἕξι σελίδες. Τὰ κείμενα αὐτὰ εἶναι:

1. Ἡρόδοτος, *Ἱστορῖαι*, Α′ (I–III)
2. Ξενοφών, *Κύρου ἀναβάσεως Α′* (1.IV.11–1.IV.13)
3. Λυσίας, *Ἀπολογία δωροδοκίας ἀπαράσημος* (21.1–21.6)
4. Πλούταρχος, *Βίοι* (Θεμιστοκλῆς, II.1–II.5)
5. Στράβων, *Γεωγραφία* (7.1.1–7.1.5)

Πρὸς μεγάλη μας ἔκπληξη τὸ TEX συλλάβισε σωστὰ ὅλες τὶς λέξεις σὲ ὅλα τὰ κείμενα.

Ἀξίζει ἀκόμα νὰ σημειώσουμε ὅτι τὸ 2ο κείμενο ποὺ χρησιμοποιήσαμε γιὰ τὴν δοκιμὴ τῶν κωδίκων συλλαβισμοῦ εἶναι αὐτὸ ποὺ χρησιμοποίησε καὶ ὁ Χαραλάμπους γιὰ τὴν δοκιμὴ τῶν δικῶν του κωδίκων [6]. Στὸ Σχῆμα 1 παρουσιάζεται τὸ ἴδιο κείμενο μὲ ὑπογραμμισμένα τὰ σημεῖα ὅπου ὁ συλλαβισμὸς διαφέρει ἀπὸ ἐκεῖνον τοῦ Χαραλάμπους [6].

Ἐν-τεῦ-θεν (*Ἐντεῦ-θεν*) ἐξ-ε-λαύ-νει στα-θμοὺς (*σταθ-μοὺς*) τρεῖς πα-ρα-σάγ-γας πεν-τε-καί-δε-κα πα-ρὰ τὸν Εὐ-φρά-την πο-τα-μόν, ὅν-τα τὸ εὖ-ρος τετ-τά-ρων στα-δί-ων· καὶ πό-λις αὐ-τό-θι ᾧ-κεῖ-το με-γά-λη καὶ εὐ-δαί-μων Θά-ψα-κος ὄ-νο-μα. Ἐν-ταῦ-θα (*Ἐνταῦ-θα*) ἔ-μει-νεν ἡ-μέ-ρας πέν-τε. Καὶ Κῦ-ρος με-τα-πεμ-ψά-με-νος τοὺς στρα-τη-γοὺς (*στρατ-η-γοὺς*) τῶν Ἑλ-λή-νων ἔ-λε-γεν ὅ-τι ἡ ὁ-δὸς ἔ-σοι-το πρὸς βα-σι-λέ-αν μέ-γαν εἰς Βα-βυ-λῶ-να· καὶ κε-λεύ-ει αὐ-τοὺς λέ-γειν ταῦ-τα τοῖς στρα-τι-ώ-ταις καὶ ἀ-να-πεί-θειν ἕ-πε-σθαι. Οἱ δὲ ποι-ή-σαν-τες ἐκ-κλη-σί-αν ἀ-πήγ-γε-λον (*ἀπ-ήγ-γε-λον*) ταῦ-τα· οἱ δὲ στρα-τι-ῶ-ται ἐ-χα-λέ-παι-νον τοῖς στρα-τη-γοῖς (*στρατ-η-γοῖς*), καὶ ἔ-φα-σαν αὐ-τοὺς πά-λαι ταῦτ' εἰ-δό-τας κρύ-πτειν, καὶ οὐκ ἔ-φα-σαν ἰ-έ-ναι, ἐ-ὰν μή τις αὐ-τοῖς χρή-μα-τα δι-δῷ, ὥσ-περ (*ὥ-σπερ*) τοῖς προ-τέ-ροις με-τὰ Κύ-ρου ἀ-να-βᾶ-σι [πα-ρὰ τὸν πα-τέ-ρα τοῦ Κύ-ρου], καὶ ταῦ-τα οὐκ ἐ-πὶ μά-χην ἰ-όν-των, ἀλ-λὰ κα-λοῦν-τος τοῦ πα-τρὸς Κῦ-ρον. Ταῦ-τα οἱ στρα-τη-γοὶ (*στρατ-ηγοὶ*) Κύ-ρῳ ἀ-πήγ-γε-λον (*ἀπ-ήγ-γε-λον*). Ὁ δ' ὑ-πέ-σχε-το (*ὑπ-έ-σχε-το*) ἀν-δρὶ ἑ-κά-στῳ δώ-σειν πέν-τε ἀρ-γυ-ρί-ου μνᾶς, ἐ-πὰν εἰς Βα-βυ-λῶ-να ἥ-κω-σι, καὶ τὸν μι-σθὸν ἐν-τε-λῆ μέ-χρι ἂν κα-τα-στή-σῃ τοὺς Ἕλ-λη-νας εἰς Ἰ-ω-νί-αν πά-λιν. Τὸ μὲν δὴ πο-λὺ τοῦ Ἑλ-λη-νι-κοῦ οὕ-τως ἐ-πεί-σθη.

Σχῆμα 1: Τὸ 2ο κείμενο ποὺ χρησιμοποιήθηκε κατὰ τὴν δοκιμὴ τῶν κωδίκων συλλαβισμοῦ. Οἱ ὑπογραμμισμένες λέξεις εἶναι ἔχουν συλλαβισθεῖ διαφορετικὰ μὲ τοὺς κώδικες τοῦ Χαραλάμπους [6]. Ὁ συλλαβισμὸς σύμφωνα μὲ τοὺς κώδικες τοῦ Χαραλάμπους δίνεται ἐντὸς παρενθέσεων.

## 4. Πιθανὰ λάθη συλλαβισμοῦ

Τὰ περισσότερα ἀναμενόμενα λάθη συλλαβισμοῦ σχετίζονται σχεδὸν ἀποκλειστικὰ μὲ τὸν συλλαβισμὸ σύνθετων λέξεων. Ἴσως μερικοὶ νὰ ἀπογοητευθοῦν ποὺ ὁ *στρατηγὸς* συλλαβίζεται ὡς *στρα-τη-γὸς* καὶ ὄχι ὡς *στρατ-η-γός*. Ὅμως, ὅπως ἀναφέραμε καὶ στὴν Εἰσαγωγή, οἱ δικοί μας κώδικες συλλαβισμοῦ τῶν ἀρχαίων ἑλληνικῶν γράφηκαν μὲ βάση τὶς ὁδηγίες τῆς Ἀκαδημίας Ἀθηνῶν καὶ ὄχι μὲ βάση τὸ *Chicago Manual of Style* [6].

Ἕνα ἄλλο σχετικὸ πρόβλημα ἔχει νὰ κάνει μὲ τὴν ἀμφισημία ὁρισμένων ὁμοιόγραπτων λέξεων. Γιὰ παράδειγμα, ἀλλιῶς συλλαβίζεται τὸ ἐπίθετο ὁ/ἡ *ἄν-ο-δος* (*α* στερ. + *ὁδός*, δηλαδὴ *ἄβατος*) καὶ ἀλλιῶς τὸ οὐσιαστικὸ ἡ *ἄ-νο-δος* (*ἄνω* + *ὁδός*, δηλαδὴ *πορεία πρὸς τὰ ἐπάνω*)· ἀλλιὼς συλλαβίζεται ἡ *προσ-κοπή* (*πρὸς* + *κόπος*, δηλαδὴ *πρόσκομμα*, *ἐμπόδιο*) καὶ ἀλλιῶς ἡ *προ-σκοπή* (*πρὸ* + *σκοπέω*, δηλαδὴ *προκατόπτευση*, *πρόβλεψη*). Τὸ TEX δὲν μπορεῖ νὰ καταλάβει τέτοιες λεπτὲς διαφορές, ἀφοῦ κάνει μόνο στοιχειοθεσία καὶ ὄχι ἀνάλυση τοῦ κειμένου.

Στὴν ποίηση, ὁρισμένες λέξεις παρουσιάζουν συνίζηση φωνηέντων ἢ διφθόγγων, δηλαδὴ δύο συνεχόμενα φωνήεντα ἢ ἕνα φωνῆεν καὶ ἕναν δίφθογγο ποὺ προφέρονται σὰν μία συλλαβή, π.χ., *χρυ-σέ͡ω*. Τὰ φωνήεντα ποὺ ἔχουν πάθει συ-

νίζηση θὰ συλλαβισθοῦν λανθασμένα ὡς δύο χωριστὲς συλλαβές, π.χ., *χρυ-σέ-ῳ*. Ἐπιπλέον, στὴν λυρικὴ ποίηση συναντοῦμε σπανίως λέξεις ποὺ περιέχουν τὸ ϝ· αὐτὲς οἱ λέξεις δὲν θὰ συλλαβισθοῦν καθόλου. Κάτι τέτοιο θὰ ἦταν δυνατὸ μόνο ἐὰν τὸ ϝ εἶχε κωδικοποιηθεῖ ὡς ἕνας συγκεκριμένος χαρακτήρας, π.χ. στὴν θέση τοῦ v ὅπως πρότεινε κάποτε ὁ Χαραλάμπους [9].

Ἴσως ἀκόμα νὰ παρουσιασθοῦν προβλήματα στὸν συλλαβισμό, ἐπειδὴ ὁ κατάλογος ἀρχικῶν μερῶν σύνθετων λέξεων, παρ' ὅτι ἐκτεταμένος, δὲν μπορεῖ νὰ εἶναι πλήρης. Παραπάνω, ἀναφέρθηκε ὅτι στὸν κατάλογο τῶν ἀρχικῶν μερῶν σύνθετων λέξεων ἔχουμε προσθέσει καὶ ὁρισμένα ριζικὰ νεώτερων λέξεων, ἔτσι ὥστε νὰ μπορεῖ κάποιος νὰ χρησιμοποίησει τοὺς ἴδιους κώδικες καὶ γιὰ τὴν στοιχειοθεσία κειμένων τῆς καθαρεύουσας. Ὡστόσο τὰ ριζικὰ ποὺ προσθέσαμε γιὰ τὴν καθαρεύουσα εἶναι πολὺ λίγα καὶ γι' αὐτὸ ἴσως νὰ παρουσιασθοῦν κάποια λάθη στὸν συλλαβισμὸ νεώτερων σύνθετων λέξεων.

Ἀναμφισβήτητα ἡ πολυμορφία τῆς ἑλληνικῆς γλώσσας μπορεῖ νὰ προκαλέσει σοβαρὰ προβλήματα συλλαβισμοῦ. Φαντασθεῖτε ὅτι θέλουμε νὰ στοιχειοθετήσουμε κάποιο παπαδιαμαντικὸ κείμενο. Ποιούς κώδικες συλλαβισμοῦ θὰ πρέπει νὰ χρησιμοποιήσουμε, ἐφ' ὅσον ὁ Παπαδιαμάντης ἔγραφε τὰ κείμενά του στὴν καθαρεύουσα, ἀλλὰ γιὰ τοὺς διαλόγους χρησιμοποιοῦσε δημοτική;

Κλείνοντας θὰ πρέπει νὰ ποῦμε ὅτι οἱ παρόντες κώδικες συλλαβισμοῦ ἔχουν γραφεῖ σύμφωνα μὲ τὴν κωδικοποίηση τῶν ἑλληνικῶν τοῦ TEX κατὰ Levy [10] καὶ χρησιμεύουν γιὰ στοιχειοθεσία μὲ τὸ GREEKTEX τοῦ Δρυλλεράκη [11] ἢ μὲ τὴν ἑλληνικὴ ἐπιλογὴ τοῦ **βαβελ** [12, 13]. Οἱ κώδικες ἴσως χρησιμεύσουν καὶ γιὰ ἄλλα πακέτα στοιχειοθεσίας ἑλληνικῶν κειμένων, ὅπως τὸ `ibycus` [14] καὶ τὸ `greektex` τοῦ Μοσχοβάκη [15], ἀλλὰ σὲ μιὰ τέτοια περίπτωση θὰ χρειαστεῖ νὰ γίνουν ὁρισμένες ἀλλαγές. Τέλος, εὐελπιστοῦμε ὅτι μία ἡμέρα οἱ κώδικες συλλαβισμοῦ τῶν ἀρχαίων ἑλληνικῶν θὰ βροῦν τὸν δρόμο τους ἀκόμα καὶ μέσα στὸ πρόγραμμα *Ὠμέγα*.

## 5. Ποῦ θὰ βρεῖτε τοὺς νέους κώδικες συλλαβισμοῦ

Οἱ κώδικες συλλαβισμοῦ τῶν ἀρχαίων ἑλληνικῶν περιέχονται σὲ ἕνα ἀρχεῖο μὲ τὸ ὄνομα `GRAhyph?.tex`, ὅπου ? εἶναι ὁ ἀριθμὸς τῆς τρέχουσας ἔκδοσής του (2). Μαζὶ μὲ τὸ ἀρχεῖο τῶν κωδίκων θὰ βρεῖτε καὶ ἕνα ἀρχεῖο LATEX μὲ τὸ ὄνομα `compound.ltx` ποὺ παρουσιάζει ὅλες τὶς σύνθετες λέξεις ποὺ ἀπαιτοῦν ἰδιαίτερη προσοχὴ στὸν συλλαβισμό.

Τὸ ἀρχεῖο μὲ τοὺς κώδικες μπορεῖτε νὰ τὸ λάβετε μὲ `ftp` ἀπὸ τὸν κατάλογο `pub/TeX/greek/hyphen` στὸν κόμβο τοῦ **εφτ** (`obelix.ee.duth.gr`). Θὰ τὸ βρεῖτε ἐπίσης στοὺς ἀντίστοιχους καταλόγους τοῦ CTAN.

## Εὐχαριστίες

Ὁ συγγραφέας θὰ ἤθελε νὰ εὐχαριστήσει τὸν κ. Σπύρο Κωνσταντᾶτο, λέκτορα Κλασικῶν Σπουδῶν στὸ Πανεπιστήμιο McGill τοῦ Καναδᾶ, γιὰ ὁρισμένες παρατηρήσεις του καὶ γιὰ τὴν προσφορὰ τῶν λεξικῶν τῆς βιβλιοθήκης του.

## Βιβλιογραφία

[1] Δ. Ἀ. Φιλίππου (2000), «Βελτιωμένοι κώδικες συλλαβισμοῦ πολυτονικῶν καὶ μονοτονικῶν νεοελληνικῶν κειμένων γιὰ τὸ TEX καὶ τὸ LATEX», *Τὸ Εὐτυπον*, νο. 4, σελ. 1–16.

[2] H. W. Smyth (1984), *Greek Grammar*, revised by G. M. Messing. Harvard University Press, Boston, USA, §138–141.

[3] W. W. Goodwin (1892), *A Greek Grammar*. Ginn and Company, Boston, USA, pp. 24–25.

Νεώτερη ἐπανέκδοση τοῦ ἴδιου βιβλίου [W. W. Goodwin (1988), *A Greek Grammar*, revised by C. B. Gulick. Orpheus Publishing Inc., New Rochelle, New York, USA, pp. 26–27] περιέχει λάθη στοὺς κανόνες συλλαβισμοῦ τῶν ἀρχαίων ἑλληνικῶν.

[4] Ι. Καλλιτσουνάκης (1939), «Ὀρθογραφικὸν διάγραμμα τῆς Ἀκαδημίας Ἀθηνῶν». *Πρακτικὰ Ἀκαδημίας Ἀθηνῶν*, τόμος ιδ′.

Στὸ ἄρθρο αὐτὸ γίνεται ἀναφορὰ ἀπὸ τὸν Χαραλάμπους [6].

[5] Μ. Χ. Οἰκονόμου (1987), *Γραμματικὴ τῆς Ἀρχαίας Ἑλληνικῆς*, ε′ ἔκδ. Ὀργανισμὸς Ἐκδόσεως Διδακτικῶν Βιβλίων, Ἀθήνα, §30–34.

[6] Y. Haralambous (1992), "Hyphenation patterns for ancient Greek and Latin." *TUGboat* **13**, 457–469.

[7] H. G. Liddell καὶ R. Scott (1968), *A Greek-English Lexicon*, revised by H. S. James and R. McKenzie. Oxford University Press, Claredon Press, Oxford, UK.

[8] Ι. Δρ. Σταματάκος (1972), *Λεξικὸν τῆς ἀρχαίας ἑλληνικῆς γλώσσης*. Ἐκδοτικὸς Ὀργανισμὸς «Ὁ Φοίνιξ» ΕΠΕ, Αθήνα.

[9] Y. Haralambous (1991), "On TEX and Greek..." *TUGboat* **12**, 224–226.

[10] S. Levy (1988), "Using Greek fonts with TEX." *TUGboat* **9**, 20-24.

[11] K. I. Δρυλλεράκης (1994), GREEKTEX, vers. 4.0$\alpha$. `ftp://laotzu.doc.ic.ac.uk/pub/tex`.

[12] C. Beccari καὶ A. Syropoulos (1998), "New Greek fonts and the `greek` option of the babel package." *TUGboat* **19**, 419–425.

[13] J. Braams (1998), babel, vers. 3.6. CTAN: `tex-archive/macros/latex/required/babel`.

[14] P. A. MacKay (1995), `ibygrk`. CTAN: `tex-archive/language/greek/ibygrk`.

[15] Γ. Μοσχοβάκης (1994), `greektex`. CTAN: `tex-archive/language/greek/greektex`.

# Λογοτεχνικός προγραμματισμός: Ή πώς να δημιουργείτε προγράμματα καθώς περιγράφετε το τι θέλετε να κάνει ο Η/Υ

Απόστολος Συρόπουλος

*28ης Οκτωβρίου 366*
*671 00 ΞΑΝΘΗ*
*apostolo@obelix.ee.duth.gr*

**Abstract**

In this article we present the literate prgramming methodology developed by Donald E. Knuth. In particular, we present the methodology and its advantages. Moreover, we present noweb, the literate programming tool developed by Norman Ramsey.

Κάνοντας τα πρώτα μου βήματα στο χώρο του προγραμματισμού έμαθα ότι ένα καλογραμμένο πρόγραμμα θα πρέπει πάντα να συνοδεύεται από μια καλή επεξήγηση της λειτουργίας του. Αυτό, συνήθως, γίνεται με την προσθήκη σχολίων κατά τη διάρκεια γραφής του κώδικα ή, στη χειρότερη των περιπτώσεων, αφού τελειώσει η διαδικασία αυτή. Επιπλέον, κάποιος αναλαμβάνει να γράψει ένα τεχνικό κείμενο το οποίο περιγράφει γενικά τον τρόπο λειτουργίας του προγράμματος και το οποίο αφορά μόνο προγραμματιστές. Προφανώς θα ήταν ευχής έργον να υπάρχει ένας τρόπος ανάπτυξης προγράμματος όπου όλες οι παραπάνω διαδικασίες θα γινόταν ταυτόχρονα και με τον καλύτερο δυνατό τρόπο. Αν πιστεύετε ότι αυτό δεν είναι δυνατό, τότε σίγουρα δεν έχετε ακούσει τίποτα για τον λογοτεχνικό προγραμματισμό.[1]

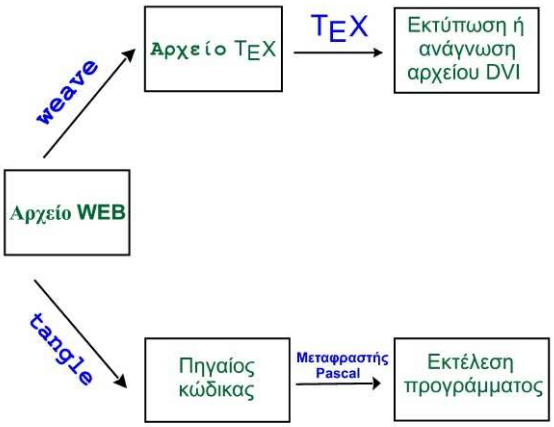
[1] Ο όρος *λογοτεχνικός προγραμματισμός* αποτελεί μετάφραση του αγγλικού όρου literate programming. Στην ορολογία των ανθρωπιστικών σπουδών, ο όρος literate μεταφράζεται ως *εγγράμματος* (-*η*, -*ο*) και ο όρος literacy ως *εγγραμματοσύνη*. Όμως το να μιλούμε για *εγγράμματο προγραμματισμό* θα ακουγόταν κάπως παράξενο στα αυτιά των προγραμματιστών, γι' αυτό και επικράτησε ο όρος *λογοτεχνικός προγραμματισμός*. Αν κάποιος άλλος έχει να προτείνει έναν καλύτερο όρο, ευχαρίστως να τον συζητήσουμε.

Ο λογοτεχνικός προγραμματισμός είναι μια μεθοδολογία ανάπτυξης προγράμματος. Η καινοτομία του λογοτεχνικού προγραμματισμού βρίσκεται στον τρόπο αντιμετώπισης του προγραμματισμού ως καθημερινή δραστηριότητα. Έτσι, αντί να φανταζόμαστε ότι η βασική μας αποστολή είναι να καθοδηγούμε τον Η/Υ για το τι πρέπει να κάνει, επικεντρώνουμε στο να επεξηγούμε σε ανθρώπους το τι θέλουμε να κάνει ο Η/Υ. Κατά αυτό τον τρόπο ο προγραμματιστής γίνεται ένας χρονογράφος, του οποίου η βασική επιδίωξη είναι η άψογη επεξήγηση και η τελειότητα του στυλ. Ο λογοτεχνικός προγραμματιστής διαλέγει προσεκτικά τα ονόματα των μεταβλητών και εξηγεί την χρήση κάθε μιας. Ο λογοτεχνικός προγραμματιστής μοχθεί για την δημιουργία ενός προγράμματος το οποίο είναι κατανοητό, εξαιτίας του γεγονότος ότι οι διάφορες έννοιες έχουν εισαχθεί κατά τέτοιο τρόπο ώστε να γίνονται κατανοητές από τους αναγνώστες του.

Η μεθοδολογία του λογοτεχνικού προγραμματισμού αποτελεί εφεύρεση του γνωστού καθηγητή της επιστήμης των Η/Υ Donald Knuth. Ο λογοτεχνικός προγραμματισμός εφευρέθηκε κατά την διάρκεια σχεδιασμού της δεύτερης έκδοσης του TEX και του METAFONT, δύο πασίγνωστων προγραμμάτων τα οποία σχεδίασε ο Κνυτη. Ο εφευρέτης αυτής της μεθοδολογίας προγραμματισμού, πίστευε ότι επειδή ο προγραμματισμός αποτελεί μια καθαρά προσωπική δραστηριότητα, αυτό που για αυτόν είχε τρομακτικά αποτελέσματα, ίσως για κάποιους άλλους να μην είχε τα προσδοκώμενα αποτελέσματα. Ο χρόνος έδειξε ότι η μεθοδολογία αυτή επιφέρει τα προσδοκώμενα αποτελέσματα, ανεξάρτητα των όποιων ιδιαιτέρων χαρακτηριστικών οποιοδήποτε προγραμματιστή. Μόλις κανείς καταπιαστεί με τον σχεδιασμό ενός προγράμματος χρησιμοποιώντας τη μεθοδολογία του λογοτεχνικού προγραμματισμού, αργά ή γρήγορα ανακαλύπτει πτυχές του προβλήματος αλλά και της λύσης που αρχικά ούτε καν είχαν περάσει από το μυαλό του. Αυτό το γεγονός κάνει το λογοτεχνικό πρόγραμμα ασύγκριτα καλύτερο από κάποιο αντίστοιχο που αναπτύχθηκε με κάποια συμβατική μέθοδο. Τούτη η διαπίστωση κάνει και τους πιο σκεπτικούς να υιοθετήσουν αυτή τη μεθοδολογία για την ανάπτυξη των προγραμμάτων των.

Στην αρχική του μορφή ένα λογοτεχνικό πρόγραμμα αποτελείται από «ανάμικτες» ενότητες κώδικα στη γλώσσα προγραμματισμού Pascal και κειμένου στα αγγλικά. Η κάθε ενότητα περιέχει τον κώδικα που λύνει κάποιο επιμέρους πρόβλημα αλλά και την επεξήγηση της λειτουργίας του. Όλες μαζί οι ενότητες κώδικα συνθέτουν το τελικό πρόγραμμα. Ο αναγνώστης ίσως ανακαλύπτει κάποια σχέση μεταξύ του λογοτεχνικού προγραμματισμού και του δομημένου προγραμματισμού. Η σχέση δεν είναι τυχαία, αλλά είναι γεγονός ότι ο λογοτεχνικός προγραμματισμός είναι μια πιο γενική μεθοδολογία, αφού μπορεί να εφαρμοστεί σε προγραμματιστικά στυλ όπου αποτυγχάνει ο δομημένος προγραμματισμός. Κάθε λογοτεχνικό πρόγραμμα ονομάζεται WEB (ιστός) και φυλάσσεται σε κάποιο αρχείο με προέκταση `web`. Ένα ειδικό πρόγραμμα, το tangle (ξεμπλέκω), αναλαμβάνει να εξάγει τον κώδικα Pascal από το WEB, ενώ ένα άλλο πρόγραμμα,

**Σχήμα 1**




Διττή χρήση αρχείου WEB

το weave (υφαίνω), παράγει ένα αρχείο TEX το οποίο περιέχει την τεκμηρίωση (documentation) του προγράμματος.

Στο Σχήμα 1 φαίνεται η διττή χρήση του ενός αρχείου WEB. Έτσι λοιπόν κανείς γράφει σε ένα αρχείο, π.χ., `foo.web`, το λογοτεχνικό του πρόγραμμα και με την εντολή `tangle foo.web` παίρνει το αρχείο `foo.p` το οποίο περιέχει τον κώδικα Pascal. Στη συνέχεια το πρόγραμμα αυτό μπορεί να μεταφραστεί σε γλώσσα μηχανής με τον αγαπημένο σας compiler (αποτελεί γνώμη μου ότι ο όρος «μεταφραστής» δεν αποδίδει το ακριβές νόημα της λέξεως compiler, ενώ δυστυχώς δεν έχει πέσει στην αντίληψη μου κάποια άλλη σωστότερη απόδοσή της). Από την άλλη με την εντολή `weave foo.web` παράγεται το αρχείο `foo.tex` το οποίο μπορούμε να τροφοδοτήσουμε στο TEX και στην συνέχεια μπορούμε να τυπώσουμε το στοιχειοθετημένο αποτέλεσμα που περιέχεται στο αρχείο `foo.dvi`. Γίνεται λοιπόν κατανοητό ότι γενικά το πρόγραμμα tangle παράγει πηγαίο κώδικα σε κάποια γλώσσα προγραμματισμού, ενώ το πρόγραμμα weave παράγει ένα αρχείο σε κάποιο γλώσσα μορφοποίησης κειμένου. Γιατί όμως ο Knuth διάλεξε την Pascal και το TEX; Όμως ο λόγος για τον οποίο ο Knuth διάλεξε το TEX είναι προφανής, ενώ ο λόγος για τον οποίο διάλεξε την Pascal είναι ότι την εποχή εκείνη (δεκαετία του 1980) η γλώσσα αυτή διδάσκονταν σε όλα σχεδόν τα πανεπιστήμια του κόσμου (εκτός των ελληνικών...) και έτσι ο πολύς κόσμος θα μπορούσε να χρησιμοποιήσει τα νέα εργαλεία. Αξίζει βέβαια να σημειώσουμε ότι η Pascal δεν αποτέλεσε ποτέ την αγαπημένη γλώσσα προγραμματισμού του Knuth, απλά γιατί δεν παρείχε εκείνες τις δυνατότητες που κάνουν τον προγραμματισμό συστημάτων απλή υπόθεση. Ενώ η μη υποστήριξη χρήσης αλλά και δημιουργίας μακροεντο-

λών αντισταθμίστηκε από την δυνατότητα του προγράμματος tangle να χειρίζεται μακροεντολές.

Αν και το σύστημα WEB και γενικότερα ο λογοτεχνικός προγραμματισμός βρήκε γρήγορα πολλούς και πιστούς φίλους, εντούτοις σύντομα έγινε κατανοητό ότι η μεγάλη εξάρτηση του συστήματος από την Pascal έκανε τουλάχιστον δυσχερής την εξάπλωση της ιδέας του λογοτεχνικού προγραμματισμού. Έτσι πολύ σύντομα έκαναν την εμφάνιση τους αρκετά συστήματα WEB για διάφορες γλώσσες προγραμματισμού, όπως η Ada, η Modula-2, η FORTRAN, η APL, η Scheme, κ.λπ. Ειδικά για την γλώσσα προγραμματισμού C (αλλά και την C++), ο Silvio Levy σε συνεργασία με το Donald Knuth δημιούργησαν το σύστημα CWEB, το οποίο αποτελεί κατά τον Knuth το τέλειο προγραμματιστικό εργαλείο. Όμως ακόμη και το CWEB είναι ένα εργαλείο εξαρτημένο από μία γλώσσα προγραμματισμού αλλά και μία γλώσσα μορφοποίησης κειμένου. Για τον λόγο αυτό ο Norman Ramsey σχεδίασε το σύστημα SpiderWEB το οποίο επιτρέπει στον χρήστη του να δημιουργήσει το δικό του WEB για την γλώσσα προγραμματισμού της επιλογής του (ως γλώσσα μορφοποίησης παραμένει το TEX). Δηλαδή, για κάθε γλώσσα προγραμματισμού μπορούμε να δημιουργήσουμε τα αντίστοιχα προγράμματα weave και tangle. Επειδή όμως και το SpiderWEB είχε τα μειονεκτήματα του, σχεδόν ταυτόχρονα ο Preston Briggs και ο Norman Ramsey δημιούργησαν τα συστήματα nuweb και noweb αντίστοιχα. Το nuweb χρησιμοποιεί το LATEX ως γλώσσα μορφοποίησης κειμένου, ενώ λειτουργεί με οποιαδήποτε γλώσσα προγραμματισμού. Το noweb κάνει ένα βήμα παραπέρα αφού μεν λειτουργεί με οποιαδήποτε γλώσσα προγραμματισμού, ενώ μπορεί και παράγει έξοδο στις γλώσσες μορφοποίησης κειμένου TEX, LATEX troff και HTML. Επειδή το noweb είναι το εργαλείο που έχει τραβήξει την περισσότερη προσοχή των μελών της κοινότητας των λογοτεχνικών προγραμματιστών, θα περιγράψουμε τον τρόπο χρήσης του εργαλείου.

Ένα αρχείο noweb αποτελείται από «ενότητες» οι οποίες μπορούν να εμφανίζονται με όποια σειρά εμείς θέλουμε και μπορούν να περιέχουν τεκμηρίωση ή/και κώδικα. Το τμήμα της τεκμηρίωσης πρέπει να ξεκινάει με το σύμβολο `@` στην πρώτη στήλη και να ακολουθείτε από ένα τουλάχιστον κενό ή μια αλλαγή γραμμής. Το τμήμα κώδικα ξεκινάει με `<<'onoma tm'hmatos>>=` το οποίο θα πρέπει να βρίσκεται σε μια γραμμή από μόνο του και να ξεκινάει από την πρώτη στήλη. Ως όνομα τμήματος μπορεί να είναι μια λέξη ή και μία ολόκληρη φράση. Μια ενότητα τερματίζεται με την αρχή μιας νέας ενότητας.

Τα τμήματα τεκμηρίωσης περιέχουν κείμενο το οποίο αγνοείται από το αντίστοιχο πρόγραμμα tangle του noweb που ονομάζεται notangle, ενώ το αντίστοιχο πρόγραμμα weave του noweb που ονομάζεται noweave το εμφανίζει στην οθόνη του H/Y αφού προηγούμενα το περάσει από ένα φίλτρο ώστε να το πάρουμε στη μορφή που επιθυμούμε. Το noweb, όπως αναφέραμε, μπορεί να συνεργαστεί με το LATEX το plain TEX, το troff ή την HTML. Με το plain TEX εισαγάγει μια αναφορά σε ένα μακροπακέτο TEX, το nwmac, το οποίο ορίζει εντολές όπως η

`\chapter`, για κεφάλαια, και η `\section`, για ενότητες. Με το LaTeX χρησιμοποιεί το πακέτο noweb, ενώ με την HTML τα πράγματα είναι απλά αφού παράγει ένα ολοκληρωμένο αρχείο χωρίς αναφορές σε τίποτα.

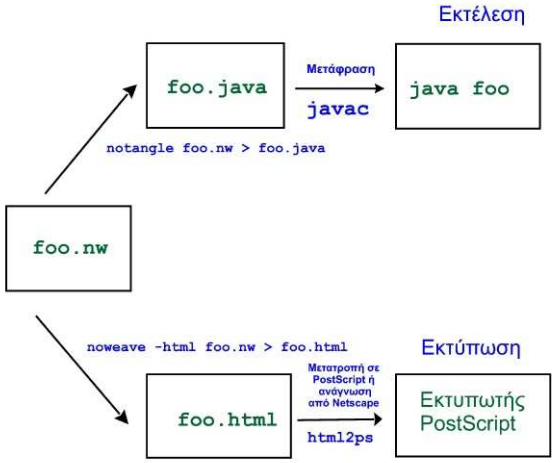
Τα τμήματα κώδικα περιέχουν πηγαίο κώδικα και αναφορές σε άλλα τμήματα κώδικα άλλων ενοτήτων. Η συγγραφή ενός τμήματος κώδικα είναι κάτι ανάλογο με τη δημιουργία μιας μακροεντολής: το notangle εξ ορισμού εξάγει ένα πρόγραμμα με το να αναπτύξει ένα βασικό τμήμα κώδικα, συνήθως με όνομα `<<*>>`. Αυτό περιέχει αναφορές σε άλλα τμήματα κώδικα, τα οποία με την σειρά τους περιέχουν άλλες αναφορές· όλες αυτές οι αναφορές αναπτύσσονται πλήρως και έτσι τελικά παράγεται και εκτυπώνεται στην οθόνη του Η/Υ ο κώδικας του προγράμματος μας. Το καλό με το notangle είναι ότι δεν αλλοιώνει τη μορφή της γραφής του κώδικα, όπως κάνουν τα παραδοσιακά συστήματα. Αν σε ένα τμήμα τεκμηρίωσης θέλουμε να γράψουμε ένα μικρό τμήμα κώδικα, θα πρέπει να το γράψουμε μεταξύ των συμβόλων `[[` και `]]`, π.χ.: `[[i++]]` ή `[[a[6]]]`. Αν θέλουμε να έχουμε το σύμβολο `@` στη πρώτη στήλη μιας γραμμής κώδικα, τότε απλά γράφουμε δύο φορές το σύμβολο, δηλ. `@@`. Αν θέλουμε τα σύμβολα `<<` και `>>` να χάσουν την προκαθορισμένη σημασία των, απλά βάζουμε μπροστά των το σύμβολο `@` (π.χ., `@<<`). Αν μία γραμμή έχει τη μορφή `@ %def` ονόματα, αυτό σημαίνει ότι η προηγούμενη ενότητα όριζε τα ονόματα που εμφανίζονται στη λίστα των ονομάτων. Τα ονόματα χωρίζονται με κενούς χαρακτήρες.

Τα δύο προγράμματα, δηλαδή το noweave και το notangle, δέχονται μια σειρά από ορίσματα γραμμής εντολών. Αναφέρουμε τα δύο πιο σπουδαία για το notangle πρώτα.

1. Αν θέλουμε να μετατρέψουμε τα τμήματα τεκμηρίωσης σε σχόλια, τότε απλά γράφουμε τον τύπο σχολίων που υποστηρίζει η γλώσσα μας. Το σύστημα αναγνωρίζει σχόλια της μορφής των παρακάτω γλωσσών (ορισμάτων): AWK (`-awk`), C (`-c`), ICON (`-icon`), SML (`-ml`), Modula-3 (`-m3`), Pascal (`-pascal`), FORTRAN77 (`-f77`) και FORTRAN90 (`-f90`). Εξ ορισμού θεωρεί ότι τα σχόλια είναι της μορφής που αναγνωρίζει η C.

2. Αν για κάποιο λόγο έχουμε κάποιο λάθος καλό είναι να μπορούμε βλέποντας το αρχείο του κώδικα να διορθώνουμε αμέσως το λογοτεχνικό πρόγραμμα. Η σύνδεση αυτή γίνεται δυνατή με το όρισμα `-L` και την παράμετρό της, η οποία καθορίζει την μορφή ενός σχολίου, που όταν εμφανίζεται στο αρχείο κώδικα δηλώνει την αντίστοιχη γραμμή στο αρχείο του λογοτεχνικού προγράμματος. Η παράμετρος είναι ένα κορδόνι στο οποίο τα σύμβολα `%F`, `%L` και `%N` δηλώνουν το όνομα του πηγαίου αρχείου, τον αριθμό γραμμής και αλλαγή γραμμής αντίστοιχα. Το σύμβολο `%%` δηλώνει απλά το σύμβολο «επί τοις εκατό». Για παράδειγμα για την Java η παράμετρος μπορεί να είναι `//line %L "%F" %N`, ενώ για την Pascal μπορεί να είναι `(* line %L "%F" *)%N`.

**Σχήμα 2**




Διττή χρήση αρχείου `noweb`

Στην περίπτωση του noweave τα πιο σημαντικά ορίσματα καθορίζουν το είδος του παραγόμενου αρχείου, καθώς και κάποια επιπλέον χαρακτηριστικά του. Αν δεν προκαθορίσουμε το είδος του αρχείου που θέλουμε να δημιουργήσει το πρόγραμμα, αυτό δημιουργεί αρχείο LaTeX. Αλλιώς, μπορούμε να δηλώσουμε ότι επιθυμούμε τη παραγωγή αρχείου plain TeX με την παράμετρο `-tex`, αρχείου HTML με την παράμετρο `-html` ή αρχείου troff με την παράμετρο `-troff`. Αν επιθυμούμε τα τμήματα κώδικα να είναι σε HTML και τα τμήματα κειμένου σε LaTeX ώστε να μπορούμε να δημιουργήσουμε άψογα αισθητικά αρχεία HTML με το πρόγραμμα LaTeX2html, χρησιμοποιούμε την παράμετρο `-\LaTeX+html`. Τέλος, είναι δυνατή η δημιουργία ευρετηρίου ενοτήτων και μεταβλητών με την παράμετρο `-index`.

Στο Σχήμα 2 φαίνεται ένα τυπικό παράδειγμα της διττής χρήσης ενός αρχείου noweb.

Ο λογοτεχνικός προγραμματισμός αναμφισβήτητα είναι μια καινοτόμα μεθοδολογία προγραμματισμού η οποία αυξάνει την παραγωγικότητα μιας εταιρείας, αφού τεκμηρίωση και κώδικας αναπτύσσονται ταυτόχρονα με τον καλύτερο δυνατό τρόπο. Παράλληλα, έχει αποδειχθεί ότι ένας προγραμματιστής που αναπτύσσει προγράμματα με την μέθοδο του λογοτεχνικού προγραμματισμού, είναι πολύ πιο παραγωγικός από τον παραδοσιακό συνάδελφό του. Επιπλέον, ο λογοτεχνικός προγραμματισμός ως προγραμματιστική μεθοδολογία έχει βρει τη θέση του στα προγράμματα σπουδών πολλών τμημάτων Πληροφορικής, με πολύ ενθαρρυντικά αποτελέσματα. Λόγω του ότι ακόμη τα προγράμματα δεν αναπτύσσονται ως λογοτεχνικά προγράμματα, κυρίως λόγω άγνοιας, κάποιος ίσως αναρωτηθεί για το

αν είναι δυνατό να μετατραπεί ένα απλό πρόγραμμα σε λογοτεχνικό. Μια απάντηση στο πρόβλημα αυτό δίνει μια απλή και συνάμα αποδοτική μέθοδος, η μέθοδος «Spaniel», προτάθηκε από τον γράφοντα και τον Νίκο Χατζηγεωργίου/[2]

Υπάρχουν πολλά παραδείγματα χρήσης του λογοτεχνικού προγραμματισμού, για την ανάπτυξη πολύπλοκων προγραμμάτων. Αρχικά, ο λογοτεχνικός προγραμματισμός χρησιμοποιήθηκε για την ανάπτυξη της έκδοσης 2 του METAFONT και της έκδοσης 3 του TEX, δύο πολύ γνωστών προγραμμάτων. Παράλληλα, πολλά συστήματα και εργαλεία, σχετικά με την ψηφιακή στοιχειοθεσία αναπτύχθηκαν ως λογοτεχνικά προγράμματα, όπως για παράδειγμα το METAPOST, το Ωμέγα, κ.ά. Επίσης, ο Norman Ramsey ανέπτυξε ένα λογοτεχνικό πρόγραμμα 33000 περίπου γραμμών, εκ των οποίων οι 15000 γραμμές είναι τεκμηρίωσή (ως γλώσσα προγραμματισμού χρησιμοποιήθηκε η Ada). Επίσης ένας πολύ γνωστός μεταφραστής της γλώσσας C, ο lcc, αποτελεί ένα ακόμη τεράστιο λογοτεχνικό πρόγραμμα το οποίο μάλιστα εκδόθηκε σε μορφή βιβλίου από την Addisson-Wesley (βλέπε `http://www.cs.princeton.edu/software/lcc/`). Ακόμη, ο μεταφραστής μιας ειδικής γλώσσας σχεδιασμένης να εκφράζει αλγόριθμούς νευρωνικών δικτύων που μαθαίνουν, της CuPit, είναι ένα εξαιρετικό λογοτεχνικό πρόγραμμα (βλέπε `http://wwwipd.ira.uka.de/~hopp/cupit.html`). Ακόμη μπορείτε να βρείτε δεκάδες προγράμματα σε άρθρα αλλά και ιστοσελίδες, π.χ. `http://obelix.ee.duth.gr/~apostolo`.

Αν κάποιος επιθυμεί να μάθει σχετικά πράγματα για τον λογοτεχνικό προγραμματισμό γενικότερα, αλλά και το noweb ειδικότερα, μπορεί να συμβουλευθεί κάποιο από τα παρακάτω URL. Αν ενδιαφέρεστε για κάποιο βιβλίο σχετικά με τον λογοτεχνικό προγραμματισμό, τότε θα πρέπει να ρίξετε μια ματιά στην παρακάτω σελίδα στο URL: `http://www.loria.fr/services/tex/english/texbib.html` Αν απλά ενδιαφέρεστε να μάθετε για άρθρα σχετικά με τον λογοτεχνικό προγραμματισμό, τότε αξίζει να ρίξετε μια ματιά στην σελίδα που ετοίμασε ο Nelson Beebe στο URL `http://www.math.utah.edu/pub/tex/bib/litprog.html` Τέλος, μπορείτε να μάθετε ότι θέλετε σχετικά με το noweb από την σελίδα του στο URL: `http://www.cs.virginia.edu/~nr/noweb/`.

Δημοσιεύτηκε στο τεύχος
Ιουλίου-Αυγούστου 1999 του
περιοδικού *RAM*.

[2] N. Hatzigeorgiu, A. Syropoulos (1998), "Literate programming and the 'Spaniel' Method." *SIGPLAN Notices*, vol. 33, no. 12, pp. 52–56.

# Package ACCENTBX: some problems with accents in TEX and how to solve them

A.S. Berdnikov

*Institute of Analytical Instrumentation*
*Rizskii pr. 26*
*198103 St.Petersburg*
*Russia*
`berd@ianin.spb.su` *or*
`berdn@ptslab.ioffe.rssi.ru`

**Abstract**

Weak points of TEX internal command `\accent` are investigated. It appears that some features of `\accent` prevent to construct specific letters which require several accents simultaneously. Special macro which emulates the work of `\accent`, additional macro for fine tuning of accents, if-then-else structures which enable to put correct accents depending on uppercase/lowercase state and font family conditions, etc., are described. Closely related to it are the accents for I/i and J/j which are different for uppercase and lowercase letters, and similar problem occurs for so-called *title letters*. The whole package is coded in such a way that in works for *Plain*, $\mathcal{AMS}$TEX, etc., as well, not only for LATEX $2_\varepsilon$.

## 1. Introduction

The nearly inevitable feature of cyrillic encodings is that some cyrillic letters (for example, the majority of accented letters) should be assembled from pieces — accents, modifiers, composite elements, etc. There are the cases (not too rare) where *several* elements should be attached to the same base character. The typical examples are the stresses over accented letters, some specific characters from Nivh language, multi-level accents specific for Saam language, stroke-type modifiers added to some letters, and so on — some of these constructions are shown here:

$$\acute{\ddot{\text{Е}}},\qquad \acute{\text{И}},\qquad \acute{\text{Ғ}},\qquad \dot{\text{Ҳ}},\qquad \bar{\ddot{\text{О}}},\qquad \overline{\ddot{\text{Ə}}\text{А}},\qquad \ldots$$

In addition there are the cases where the accent is positioned *between* two characters — for example, in German [1] there is the abbreviation -*bǧ* which substitutes -*burg* when necessary[1]. Such effect could be simulated by a special invisible character with zero width and corresponding height as it is described in the documentation for DC/EC fonts [1], but invisible rule with zero width or `\phantom` command are better for this purpose — unfortunately `\accent` cannot place accents over them.

Another problem arises when we need the accent above the letters I/i or J/j — these letters contain dots for the lowercase form but do not contain it for the uppercase form. It is not a problem when there is no automatic uppercase and lowercase conversion of the user defined text since *dottless* lowercase forms are available in Computer Modern typefaces — but, for example, the Khakassian uppercase 'İ' could be unexpectedly transformed into double-dotted 'i' after `\lowercase` used somethere:

$$\backslash.\mathtt{I} = \dot{\mathrm{I}} \quad \mathtt{\backslash uppercase\{\backslash.I\}} = \dot{\mathrm{I}} \quad \mathtt{\backslash lowercase\{\backslash.I\}} = \mathrm{i}$$

Summary: the primitive `\accent` command cannot be used for such constructions (it cannot analyze uppercase/lowercase characters, it puts accents only over the letters and atomic symbols ignoring the composite characters like `\hbox` or `\vrule` and it cannot put accent over already accented characters). To overcome these restrictions and to support inside one encoding as much cyrillic writing systems as possible, it is necessary to simulate the work of the command `\accent` by some other TEX tools.

Finally, there is some problem which does not relate directly to accents — This is the problem of so-called *title letters* which are not supported by TEX when the commands `\lowercase` and `\uppercase` work. Although it is not the problem with `\accent` (and hence corresponding macro are not included into ACCENTBX), it closely relates to the problem with accented I/i and J/j — so, this problem and its solution is discussed in section 10.1.

---

1 In first version of this paper published in Proceedings of EuroTEX-97 there was an error: it states that sometimes in Serbian language the accents *acute*, *grave*, *double-grave* and *frown* (reverted breve) which are the specific Serbian stresses, may be positioned in a similar manner: бр̏вно, мр̀твац, мр̏ква, зр̑тва, ... More detailed investigation shows that in such cases corresponding stresses are placed strictly over the letter 'р'. But in Church Slavonic and Old Slavonic the accents *paerok* and *titlo-in-letters* placed between two letters are not a rare case.

## 2. How \accent works

Before we proceed to simulate the work of `\accent` it may be a good idea to see what particular operations are performed by this internal TEX command. Its syntaxis is specified as [2, 3]:

`\accent`⟨*8-bit number*⟩ ⟨*optional assignments*⟩ ⟨*character*⟩

*Character* is a letter over which the accent should be placed. The 8-bit number is the code from the current font which corresponds to the character used as the accent. The optional assignments between the *8-bit number* and the *character* could contain, in particular, the font switching commands which enable to take the accent defined by *8-bit number* and the accented object defined by *character* from different fonts (although it cannot contain `\setbox` assignments).

If the assignments are not followed by a *character* (where *character* is the token with `\catcode=11` or `\catcode=12`, command `\char`, TEX command created through `\chardef`, `\noboundary` command), TEX considers `\accent` as the equivalent to `\char`⟨*8-bit number*⟩ command. Otherwise the character which follows the assignment is accented by the character that corresponds to *8-bit number*. The accenting algorithm is the following[2]:

1. If the *character* has a height equal to `\fontdimen5` for the font where the accent is selected (which is `1ex` for this font), the accent is centered horizontally over the box corresponding to the *character* and no additional vertical shift is inserted.

2. If the *character* has a height unequal to `1ex` and the font is not slanted, i.e., the slant `\fontdimen1` for the font where the accent is selected is equal to `0pt`, the accent `\char`*8-bit number* is placed in a `\hbox` which is vertically shifted by the difference of the height of the *character* and `1ex`. The accent box is shifted up, if the *character* is heigher than `1ex`, otherwise it is shifted down. It is also centered horizontally to the *character* box.

3. If the *character* has a height unequal to `1ex` and the font is slanted, i.e., the value `\fontdimen1` for the font where the accent is selected is not equal to `0pt`, the accent `\char`*8-bit number* is placed in a `\hbox` and vertically shifted as described under 2.

   Horizontally the center of the accent box is displaced from the center of the *character* by the absolute amount of the horizontal shift amount

[2] This description with minor corrections belongs to Bernd Raichle and was extracted by him through investigation of the original Web code for TEX.

multiplied by the font slant `\fontdimen1` (which is given as *slant-per-pt*). It is displaced to the right, either if the box is shifted up and the font slant is positive (i.e., the glyphs of the font are slanted to the right), or if the box is shifted down and the font slant is negative, otherwise the accent box is moved to the left.

Such additional horizontal shift proportional to slant value is an essential component of TEX accenting algorithm since it enables to position the accent properly over the slanted character proportionally to its height.

The proper positioning of the accent box over the accented character is performed by the explicit kerns inserted by TEX. Kerns are inserted in such a manner that the horizontal size of the accented character is not affected. On the contrary, the vertical size of the accented character is corrected following the height of the resulting construction.

The fact that the command `\accent` inserts explicit kerns prevents TEX from using its automatic hyphenation algorithm, as it usually happens for the words with explicitly inserted kerns. It is essential that the commands for the correct alignment of the accent with respect to the accented character are inserted by TEX in the output stream at first — i.e, *before* the main character. If the italic correction command `\/` is placed after the accented character and if it is separated from the main character by some kerns, `\hbox` command, etc., it fails to define properly the *italic correction* shift.

## 3. How new `\accent` may be implemented

By definition the command `\accent` can put accents only over the atomic letters and symbols. In particular, it cannot put an accent over already accented character (at least in text mode). The other disadvantage is that accent and accented character should be taken usually from the same font. Although the syntaxis of `\accent` enables to use the font switching commands, this possibility is realized not in a comfortable mode. Usually we would like to select the accent from a special font, and the main character — from the current font. Inspite of it the `\accent` syntaxis forces to store and to reconstruct explicitly the current font status when the accent is selected not from the current font.

Some simple macro emulating the work of `\accent` and enabling to overcome its restrictions are suggested in [4][3]:

```
\def\ifnnull#1{\def\inner{#1}\ifx\inner\empty\else}
```

[3] The original commands are modified a little to fit the purposes of this paper.

```
\def\genaccent#1#2#3{%
   \leavevmode\setbox0=\hbox{#3}%
   \vbox{\offinterlineskip
         \ifnnull{#1}\hbox to \wd0{\hss#1\hss}\kern 0.2ex\fi
         \vbox to \ht0{\copy0\vss}%
         \vtop{\null\vbox to \dp0{\vss}%
           \ifnnull{#2}\kern 0.2ex \hbox to \wd0{\hss#2\hss}\fi
         }%
   }%
}
```

It enables to put above and below an arbitrary TEX object as many accents as you wish (defined as `\hbox` if it is necessary). It is also not a problem to select the accents used for this purpose from a separate font without taking care about the current font status. Unfortunately these commands do not take into account the fine accent tuning for slanted fonts, and they also break down the work of italic correction `\/` (the main character is deeply hidden inside `\vbox`).

This scheme could be modified. Suppose the accent is defined as the first parameter of some macro, and the main symbol to be accented – as its second parameter. The following commands

```
\setbox0=\hbox{#2}
\setbox2=\hbox to \wd0{\hss#1\hss}
\raise\ht0\box2\kern-\wd0 \unhbox0
```

do the necessary work:[4]

— put the accent above the main character,

— align it properly in horizontal direction (at least for straight fonts),

— does not change the width of the main character while the height of the composite construction is corrected properly,

— leaves the main character as the last object in the output stream so that the italic correction command `\/` can have a free access to it,

— enables to use arbitrary TEX constructions for accents and accented characters,

[4] Here the command `\unhbox` is used instead of `#2` following the recommendations by Bernd Raichle to avoid the side effects if there are assignments inside the argument `#2`.

| Case 1: | Case 2: |
|---|---|
| `\dimenA=...`<br>`\dimenB=...`<br>`\dimenC=\dimenA`<br>`\countX=\dimenB`<br>`\multiply\dimenC by \countX`<br>`\divide\dimenC by 65536` | `\dimenA=...`<br>`\dimenB=...`<br>`\dimenC=\dimenA`<br>`\countX=\dimenB`<br>`\divide\dimenC by 65536`<br>`\multiply\dimenC by \countX` |

Figure 1: How to multiply $2pt \times 2pt \to 4pt$ in TEX

— conserves the kerning with the characters followed after the accented character (unfortunately the kerning with the *previous* character is destroyed irreversibly).

The only thing to do is to add the fine horizontal tuning which is not a problem as soon as we can calculate it. But the calculation of the horizontal shift necessary for slanted fonts *is* a problem and requires additional tricks! It is necessary to multiply slant value `\fontdimen1` by the vertical shift of the accent, but the problem is that TEX contains *no* primitive tools to multiply `2pt` by `2pt` and to get `4pt` as a result (at least if the multiplied values are the `\dimen` registers).

One way to do such operation is to convert one value to an integer (assuming $1pt$=65536) and perform multiplication as it is shown on Fig. 1. Unfortunately the accuracy of such algorithm is very low: in the first case the overflow could occur after `\multiply\dimenC by \countX` if `\dimenA` is too big; in the second case the result after `\divide\dimenC` by 65536 could be zero id `\dimenA` is too small. Some special macro exist [5, 6, 7, 8] which perform floating point calculations in TEX with high accuracy. Hopefully we do not need such intensive calculations. The macro `\slant@value` which converts the `\fontdimen1` specified for the current font into numerical string (like `\the` command, but without the suffix `pt` at its end) would be enough for our purpose:

```
\setbox0=\hbox{#2}
\setbox2=\hbox to \wd0{\hss#1\hss}
\dimen2=\slant@value\ht0
\kern\dimen2
   \raise\ht0\box2\kern-\wd0
\kern-\dimen2 #2
```

The macro which converts the `\fontdimen1` of the current font into text string without the suffix `pt` at its end can be coded, for example, as

```
\begingroup\catcode‘P=12 \catcode‘T=12
           \lowercase{\endgroup\def\strip@pt#1PT}{#1}}
\def\slant@value{\expandafter\strip@pt\the\fontdimen1\font}
```

(This trick is suggested by Bernd Raichle and is based on `\kslant` shown on p.375 of TEXbook.)

Other operations require a standard TEXnical work with boxws and does not cause serious problems from the designer. Unfortunately, such operations destroy automatic kerning and prevents automatic hyphenations (like ordinary command `\accent` does).

## 4. General remarks about accent commands

To create the "universal" set of accenting commants it is necessary to take into account the following factors:

1. Symbol used as accent could be (and should be) an ordinary `\hbox` — i.e., the User does not align it in advance with respect to the height of `1ex` as it is done for accent characters in CM fonts.

2. Since typical accents in CM fonts are shifted in advance verically by `1ex` and horizontally by corresponding slant correction, before we use these characters as accents we should compensate these shifts backward. Hence, the User should have at his/her disposal necessary macro.

3. Although typically accents are placed above the characters, there are cases where the modifiers are placed below the character, at the baseline of the character and over the character to make special composite objects. Hence, a variety of accenting commands is necessary.

4. Accents which are placed above the character or below the character may have additional vertical space between the accent and the character for fine tuning.

5. Accents which are the modifiers joined without space with the character may require to shift it closer to the character to make intersection well expressed.

6. Symbol like comma may require to raise it above the baseline before it can be used as the upper accent.

7. Accents can be placed before the letter at the beginning of the word, after the letter at the end of the word, or between two letters in the middle of the word, as in it explained in section 1.

8. Accents "*overline*" (*macron*) and "*underline*" may be longer than the width of one letter[5].

9. Accents may be justified with respect to the character in non-standard way — say, it can be aligned to the left corner or to the right corner of the character.

10. Pseudo accents like strokes, ets., artificially added to already existing characters, may require fine tuning of their position with respect to the body of the character. In particular, such tuning can depend on font style (italic, slanted, etc.).

11. It is necessary to take into account that the composite characters may be influenced by the commands `\uppercase` and `\lowercase` (or by equivalent LaTeX $2_\varepsilon$ commands `\MakeUppercase` and `\MakeLowercase`). Accents should be stable (since these operations change the height and the width of the character) and robust (since accented characters may be placed inside section headers, etc., where — at least for LaTeX $2_\varepsilon$ — fragile commands are not welcome).

12. Closely related to it is the problem with the accents for I/i and J/j: when converted by `\uppercase` and `\lowercase` these letters can get or loose the upper dot accent.

## 5. Some examples

Before we proceed further let me demonstrate some simple examples. (Although the commands included into ACCENTBX are rather lengthy, it is done intensionally: these commands are mainly designed for creating your own accent commands and specific characters available through macros, not for "everyday usage" inside your text.)

Suppose we would like to put a tiny letter 'a' as the accent over some character. Using the package ACCENTBX we create special command

```
\def\tA#1{\upaccent{\tiny a}{#1}}
```

[5] The Saam (Lappish) writing system is the only known up to now which has such modificators.

(`\upaccent` places one box as the accent over the second box). Now we can see how it works:

| | |
|---|---|
| $\overset{\text{a}}{\text{X}}$ = `\tA{X}` | $\overset{\text{a}}{\text{X}}$ = `\tA X` |
| $\overset{\text{a}}{\text{x}}$ = `\tA{x}` | $\overset{\text{a}}{\text{x}}$ = `\lowercase{\tA X}` |
| $\overset{\text{a}}{\mathbf{X}}$ = `\bf\tA{X}` | $\overset{a}{X}$ = `\it\tA{X}` |
| $\overset{\text{a}}{\overset{\text{a}}{\text{X}}}$ = `\tA{\tA X}` | $\overset{\text{a}}{\overset{\text{a}}{\overset{\text{a}}{\text{X}}}}$ = `\tA{\tA{\tA X}}` |
| $\overset{a}{\overset{a}{X}}$ = `\it\tA{\tA X}` | $\overset{a}{\overset{a}{\overset{a}{X}}}$ = `\it\tA{\tA{\tA X}}` |

If you would like to reproduce exactly what is typed above, you'll discover that the result is a little bit different: `\it\tA{\tA{X}}` gives $\overset{a}{\overset{a}{X}}$, not $\overset{a}{\overset{a}{X}}$. Yes, actually the previous example was defined as

```
\def\tA#1{\upaccent{\aboxsplit{\tiny a}}{#1}}
```

— since `\upaccent` just places a box over box, it is necessary to add some space around tiny 'a' to separate these boxes, and this is just what macro `\aboxsplit` does.

If you try to make ordinary TEX accents with `\upaccent`, you'll discover another essential feature:

`\upaccent{\char"13}{X}`=$\overset{\acute{}}{\text{X}}$

while you would like to have $\acute{\text{X}}$ — the space is too big because the character `\char"13` in *Computer Modern* typeface is shifted in advance to fit the letter 'x' without further adjustments. To correct this artificial shift typical for TEX accents special macro `\aboxshift` should be used:

`\upaccent{\aboxshift{\char"13}}{X}`=$\acute{\text{X}}$.

This macro subtracts the height `1ex` in vertical direction and corresponding slant correction in horizontal direction.

Similar effect is when you try to put *comma* as the accent without preliminary adjustment:

```
\def\tcomma#1{\upaccent{,}{#1}}
\tcomma{X}=
```
$\overset{,}{\text{X}}$

Since `\upaccent` alignes the boxes with respect to the baseline and comma has non-zero depth, its descender overlaps with the character to be accented. To correct it we should use this macro:

```
\def\tcomma#1{\upaccent{\aboxsplit{\aboxbase{,}}}{#1}}
\tcomma{X}=X̦
```

Macro `\aboxbase` raises the character by its depth so that the baseline of the result corresponds to its geometrical bottom, and macro `\aboxsplit` adds some extra space above and below it.

Finally, let us show how the interletter accents shown in section 1 can be created:

```
\def\burg{b\upaccent{\aboxshift{\char"15}}{\markchar}g}
St.~Peters\burg=St. Petersb̆g
```

Here macro `\markchar` creates invisible rule with zero width and the height corresponding to that of letter 'x'. Since `\upaccent` adjusts the height and the depth of the accented character but not its width, the result is the object with zero width (from TEX' point of view) placed between characters *b* and *g*.

If we would like to make our macro case-sensitive, it can be defined as

```
\def\burg{b\upaccent{\Aboxshift{\char"15}}{\markchar}g}
\def\BURG{B\upaccent{\Aboxshift{\char"15}}{\MARKCHAR}G}
```

This results to:

```
St.~Peters\burg=St. Petersb̆g
ST.~PETERS\BURG=ST. PETERSB̆G
\MakeUppercase{st.~peters\burg}=ST. PETERSB̆G
\MakeLowercase{ST.~PETERS\BURG}=st. petersb̆g
```

(Here macro `\MARKCHAR` creates the invisible rule with the height of capital letter 'X', macro `\markchar` and `\MARKCHAR` operate in such a way that they are transformed into each other after uppercasing and lowercasing, macro `\Aboxshift` is the robust version of `\aboxshift` which is not destroyed by the LATEX $2_\varepsilon$ macro `\MakeUppercase` and `\MakeLowercase`.)

## 6. Commands for accents

**Important note.** In most cases the symbol used as the accent or modifier should be adjusted in advance — see examples in the previous section and section 7 for more details.

The style file ACCENTBX contains the following commands used to make accents:

`\upaccent{`*accent*`}{`*main-symbol*`}` — put *accent* above *main-symbol*

`\dnaccent{`*accent*`}{`*main-symbol*`}` — put *accent* below *main-symbol*

`\baseaccent{`*accent*`}{`*main-symbol*`}` — put *accent* at the baseline below symbol *main-symbol*

`\nullaccent{`*accent*`}{`*main-symbol*`}` — the *accent* (i.e., some character modifier like stroke, etc., attached to the main character) is centered with respect to the height of the *main-symbol*

`\upaccentC{`*accent*`}{`*main-symbol*`}` — synonim for `\upaccent`.

`\upaccentL{`*accent*`}{`*main-symbol*`}` — put *accent* above *main-symbol* like the command `\upaccent` does, but aligns it over the left corner of the character.

`\upaccentR{`*accent*`}{`*main-symbol*`}` — put *accent* above *main-symbol* like the command `\upaccent` does, but aligns it over the right corner of the character.

`\dnaccentC{`*accent*`}{`*main-symbol*`}`, `\dnaccentL{`*accent*`}{`*main-symbol*`}`, `\dnaccentR{`*accent*`}{`*main-symbol*`}` — corresponding modifications of the command `\dnaccent`.

`\baseaccentC{`*accent*`}{`*main-symbol*`}`, `\baseaccentL{`*accent*`}{`*main-symbol*`}`, `\baseaccentR{`*accent*`}{`*main-symbol*`}` — corresponding modifications of the command `\baseaccent`.

`\nullaccentC{`*accent*`}{`*main-symbol*`}`, `\nullaccentL{`*accent*`}{`*main-symbol*`}`, `\nullaccentR{`*accent*`}{`*main-symbol*`}` — corresponding modifications of the command `\nullaccent`.

`\upaccentbar[`*height*`]{`*main-symbol*`}` — place the horizontal bar above the *character* taking into account its width (i.e., it can place the bar over more than one character). Optional parameter *height* specifies additional space which is placed around the bar. Default value of additional space is 0.2ex.

`\dnaccentbar[`*height*`]{`*main-symbol*`}` — put the horizontal bar below the *character* taking into account its width (i.e., it can place the bar under more than one character). Optional parameter *height* specifies additional space which is placed around the bar. Default value of additional space is `0.2ex`.

You can use a variety of TEX constructions as the *accent* and *main-symbol* — mainly those that can be used inside `\hbox`. In particular, the accented letter can be used as the main symbol (i.e., the multiple commands `\upaccent`, `\dnaccent`, `\baseaccent`, etc., could be used). It is essential to note that if the final symbol in a chain of accenting commands is the symbol with `\catcode` 11 or 12, the italic correction command `\/` will work correctly if it follows the whole construction.

## 7. Commands which create and correct accent characters

The special objects useful if they are used as the *accents* can be created using the following commands:

`\aboxshift[`*height*`]{`*character*`}` — lower by *height* the character (which is usually the accent from Computer Modern font family). Default value for *height* is `1ex`. Except the height the additional horizontal shift typical to slanted fonts is subtracted as well.

`\aboxbase[`*height*`]{`*character*`}` — raise the character by the specified *height* and, if there is still some depth of the result, by the residue depth. Default value for *height* is `0pt` which means that the character is just raised if some its elements are below the baseline.

`\aboxbaseline{`*character*`}` — raise or lower the character so that its bottom becomes the baseline (i.e., if the depth of the character is positive, the character is raised, if the depth is negative — the character is lowered).

`\aboxsplit[`*height*`]{`*character*`}` — insert above and below the *character* the white space with the specified *height*. Default value for the *height* is `0.2ex`.

`\aboxjoin[`*height*`]{`*character*`}` — subtract above the *character* and below it the specified *height*. Default value for the *height* is `0.1ex`.

`\aboxsplitup[`*height*`]{`*character*`}` — insert below the *character* the white space with the specified *height* (it adds white space *below* the accent

because the accent created by `\upaccent` is supposed to be *above* the character). Default value for the *height* is 0.2ex.

`\aboxsplitdn[`*height*`]{`*character*`}` — insert above the *character* the white space with the specified *height* (it adds white space *above* the accent because the accent created by `\dnaccent` is supposed to be *below* the character). Default value for the *height* is 0.2ex.

`\aboxsplitup[`*height*`]{`*character*`}` — subtract below the *character* the white space with the specified *height* (it subtracts white space *below* the accent because the accent created by `\upaccent` is supposed to be *above* the character). Default value for the *height* is 0.1ex.

`\aboxsplitdn[`*height*`]{`*character*`}` — subtract above the *character* the white space with the specified *height* (it subtracts white space *above* the accent because the accent created by `\dnaccent` is supposed to be *below* the character). Default value for the *height* is 0.1ex.

`\markchar` — make the invisible rule with the height equal to `1ex`. After `\uppercase` and `\MakeUppercase` it is converted into `\MARKCHAR`.

`\MARKCHAR` — make the invisible rule with the height equal to the height of the capital letter *X*. After `\lowercase` and `\MakeLowercase` it is converted into `\markchar`.

`\marktwochar{`$char_1$`}{`$char_2$`}` — make the invisible rule with zero width, with height equal to the maximal height of two characters and with corresponding depth. Can be substituted by `\aboxmarker` but is conserved for compatibility reasons.

`\aboxmarker{`*character*`}` — make the invisible rule with zero width, with height equal to the height of the box and with corresponding depth.

`\aboxnull{`*character*`}` — makes an artificial box with zero height and zero depth but with the same the contents as the source. The width and the baseline are conserved as well.

`\akern{`$char_1$`}{`$char_2$`}` — make the kern corresponding to the pair of characters. It is useful if you would like to reconstruct kerning destroyed by accents. Example:

| | | |
|---|---|---|
| `\upaccent{.}{A}\upaccent{.}{W}` | = | $\dot{A}\dot{W}$ |
| `\upaccent{.}{A}\akern AW\upaccent{.}{W}` | = | $\dot{A}\dot{W}$ |

`\aboxbar{`*character*`}` — create the horizontal bar which is as wide as necessary to overline or to underline the *character*. The commands `\upaccentbar` and `\dnaccentbar` described in the previous section are defined with its help as:

```
\def\upaccentbar#1{\upaccent{\aboxsplit{\aboxbar{#1}}}{#1}}
\def\dnaccentbar#1{\dnaccent{\aboxsplit{\aboxbar{#1}}}{#1}}
```

The width of the bar is scaled with respect to the nominal width of the character by factor `\accentwidthfactor` (default value 0.9). The thickness of the bar is the height of the dot selected from currect font and scaled twice by factor `\accentthickness` (default value is 0.1) — so by default the bar thickness is in 5 times less than the thickness of the dot (i.e., the character '.' of the current font).

`\aboxrule{`*width*`}` — just the same as `\aboxbar` but the width of the object to be covered by the bar is specified explicitly and the width is not scaled by `\accentwidthfactor`.

# 8. General accent `\makeaccent` and general frame `\aboxframed`,`\aboxtuning`

Macro listed above are enough for 99% of applications. But for extremely exotic accent constructions and for cases which require fine tuning of the accent and the main character ACCENTBX contains three additional commands:

— `\makeaccent` — for fine tuning of the accent and the main character (both could be some `\hbox` constructions!) with respect to each other,

— `\aboxframed` — for fine tuning of the frame and the baseline of the character used as the accent,

— `aboxtuning` — for explicit specification of frame and baseline of the box.

## 8.1. Macro `\aboxframed` and `\aboxtuning`

Macro `\aboxframed` enables to control white spaces around the character used as the accent and the position of its baseline. It has the syntaxis:

`\aboxframed[`*options*`]{`*object*`}`

where *object* is the `\hbox`-object to be adjusted, and *options* is the list of letters describing the operations:

- `r` — increase right boundary by one step,
- `u` — increase upper boundary by one step,
- `l` — increase left boundary by one step,
- `d` — increase lower boundary by one step,
- `b` — increase baseline position by one step,
- `R` — decrease right boundary by one step,
- `U` — decrease upper boundary by one step,
- `L` — decrease left boundary by one step,
- `D` — decrease lower boundary by one step,
- `B` — decrease baseline position by one step.

You can use as much lettwers as you wish, and each letter performs an independent increment or decrememt of corresponding value. The step values are defined by macro:

`\aboxframestepV` — step in vertical direction (i.e., for upper and lowerr boundaries and for baseline position), default value is `0.2ex`,

`\aboxframestepH` — step in horizontal direction (i.e., for right and left boundaries), default value is `0.08em`.

The algorithm behaves like following:

1. All five increments (right boundary, upper boundary, left boundary, lower boundary and baseline position) are collected and stored in corresponding variables.
2. The height and the depth of the object are corrected by corresponding values without any changes inside the object.
3. Left and right spaces (may be, negative) are added to the object.
4. The object is shifted (raised) by the value corresponding to the shift of the specified shift of the baseline.

To make life easier, there is the macro `\aboxtuning` which performs just the same operation in one step. Its syntaxis is:

`\aboxtuning{`$\Delta r$`,`$\Delta h$`,`$\Delta l$`,`$\Delta d$`,`$\Delta b$`}{`*object*`}`

where $\Delta r$ shows the shift of the right boundary, $\Delta h$ — the shift of the upper boundary, $\Delta l$ — the shift of the left boundary, $\Delta d$ — the shift of the lower boundary, $\Delta b$ — the shift of the baseline. Some values can be omitted (in this case they are assumed to be zero), trailing commas can be omitted also if there are no more values in a list.

## 8.2. Macro `\makeaccent`

Macro `\makeaccent` enables to control mutial alignment of the accent and the main character in more details. It has the syntaxis:

`\makeaccent[`*options*`]<`*shifts*`>{`*accent-character*`}{`*main-character*`}`

where

— *accent-character* is the `hbox`-object used as the accent,

— *main-character* is the `hbox`-object to be accented,

— *options* is the list of letters describing roughly what base points of the accent and the main character are joined when these two object are overlapped,

— *shifts* is the list of four numeric values which control fine tuning of the base points of the accent and the main character.

Macro `\makeaccent` overlaps the *main character* and the *accent character* so that their base points coinside with each other. When the accent character is lowered or raised, the corresponding slant correction is added to is horizontal shift. By default the accent and the main character are aligned so that the accent is centered above the main character and its baseline is placed on top of the main character (see below the defasult value for the parameter *options*). Parameters *options* and *shifts* are optional — they may be omitted together with surrounding angular and square braces.

Following letters can be used in *options* to specify the base points of the *accent-character* and the *main-character* which should be overlapped by TEX when composing the accented character:

`r` — right side of the main character,

`l` — left side of the main character,

`o` — center between left and right boundaries of the main character,

`t` — top of the main character,

`d` — bottom of the main character (including its depth),

`b` — baseline of the main character,

`c` — vertical center of the main character (the middle between its top and bottom boundaties including its depth),

`s` — 1/2 of the height of the main character,

`z` — 1/2 of the depth of the main character.

`R` — right side of the accent,

`L` — left side of the accent,

`O` — center between left and right boundaries of the accent,

`T` — top of the accent,

`D` — bottom of the accent (including its depth),

`B` — baseline of the accent,

`C` — vertical center of the accent,

`S` — 1/2 of the height of the accent,

`Z` — 1/2 of the depth of the accent.

For example, the combination `rt` means that the right top point of the main character is selected as the base point, and the combination `OD` means that the center of the lower boundary of the accent is selected as its base point. By default *options*=`[tBoO]` which means that the accent and the main character are aligned horizontally so that their vertical middle lines coinside, and that the top of the main character and the baseline of the accent counside as well.

In addition to rough specification of the base points of the main character and the accent character by *options*, the User may perform fine tuning of their positions using *shifts*. Namely, this parameter is the list of four values (lengths) separated by commas which are added to the coordinates of the base points specified by *options*. First value corresponds to $x$-shift of the base point of

the main character, second value — to $y$-shift of the base point of the main character, third value — to $x$-shift of the base point of the accent character, forth value — to $y$-shift of the base point of the accent character. If some values are equal to zero, they can be omitted. If there are no more non-zero values in a list, trailing commas can be omitted as well. By default parameter *shift* consistes of four zero values.

To illustrate the power of macro `\makeaccent` let us see how the accent macro described in section 6 can be specified:

```
\def\upaccentC#1#2{\makeaccent[BtOo]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\upaccentR#1#2{\makeaccent[BtRr]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\upaccentL#1#2{\makeaccent[BtLl]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\dnaccentC#1#2{\makeaccent[TdOo]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\dnaccentR#1#2{\makeaccent[TdRr]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\dnaccentL#1#2{\makeaccent[TdLl]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\baseaccentC#1#2{\makeaccent[TbOo]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\baseaccentR#1#2{\makeaccent[TbRr]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\baseaccentL#1#2{\makeaccent[TbLl]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\nullaccentC#1#2{\makeaccent[SsOo]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\nullaccentR#1#2{\makeaccent[SsRr]<0pt,0pt,0pt,0pt>{#1}{#2}}
\def\nullaccentL#1#2{\makeaccent[SsLl]<0pt,0pt,0pt,0pt>{#1}{#2}}
```

## 9. Robust versions of some commands

Since LaTeX $2_\varepsilon$ macro `\@ifnextchar` is fragile and is destroyed by some LaTeX $2_\varepsilon$ commands (for example, by `\MakeUppercase` and `\MakeLowercase`), commands with optional arguments have similar robust versions which do not have options but which are not destroyed inside LaTeX $2_\varepsilon$ stomach. The list of fragile commands is the following:

— `\aboxshift[`*height*`]{`*character*`}`:

  – `\Aboxshift{`*character*`}` assumes that *height*=1ex,

  – `\AboxShift{`*height*`}{`*character*`}` is robust.

— `\aboxbase[`*height*`]{`*character*`}`:

  – `\Aboxbase{`*character*`}` assumes that *height*=0pt,

  – `\AboxBase{`*height*`}{`*character*`}` is robust.

— `\aboxsplit[`*height*`]{`*character*`}`:

  – `\Aboxsplit{`*character*`}` assumes that *height*=0.2ex,

  - \AboxSplit{*height*}{*character*} is robust.

— \aboxsplitup[*height*]{*character*}:

  - \Aboxsplitup{*character*} assumes that *height*=0.2ex,
  - \AboxSplitup{*height*}{*character*} is robust.

— \aboxsplitdn[*height*]{*character*}:

  - \Aboxsplitdn{*character*} assumes that *height*=0.2ex,
  - \AboxSplitdn{*height*}{*character*} is robust.

— \aboxjoin[*height*]{*character*}:

  - \Aboxjoin{*character*} assumes that *height*=0.1ex,
  - \AboxJoin{*height*}{*character*} is robust.

— \aboxjoinup[*height*]{*character*}:

  - \Aboxjoinup{*character*} assumes that *height*=0.1ex,
  - \AboxJoinup{*height*}{*character*} is robust.

— \aboxjoindn[*height*]{*character*}:

  - \Aboxjoindn{*character*} assumes that *height*=0.1ex,
  - \AboxJoindn{*height*}{*character*} is robust.

— \upaccentbar[*height*]{*character*}:

  - \Upaccentbar{*character*} assumes that *height*=0.2ex,
  - \UpaccentBar{*height*}{*character*} is robust.

— \dnaccentbar[*height*]{*character*}:

  - \Dnaccentbar{*character*} assumes that *height*=0.2ex,
  - \DnaccentBar{*height*}{*character*} is robust.

— \aboxframed[*options*]{*character*}:

  - \Aboxframed{*character*} assumes that *options*=ud,
  - \AboxFramed{*options*}{*character*} is robust.

— \makeaccent[*options*]<*shifts*>{*accent*}{*character*} :

  - \Makeaccent{*accent*}{*character*} assumes that the parameters *options* and *shifts* are empty,
  - \MakeAccent{*options*}{*shifts*}{*accent*}{*character*} is robust.

## 10. Case-sensitive accents for $I$ and $J$

The letters $I$ and $J$ should be separated in a specific class among all the letters used with accents. The essential difference is that the lowercase form contains the *dot* accent while the uppercase form does not have such modifiers. To make correct accented versions of lowercase $i$ and $j$ TEX contains special characters *dotless-i* and *dottless-j*, usually defined by the commands `\i` and `\j`. As a result the correctly working command `\accent` should analyze the uppercase and lowercase forms of these letters and select proper glyph to produce correct forms of letters even after `\uppercase` and `\lowercase`.

As soon as we consider the cyrillic writing systems, the capital *Ukranian I* ($\ddot{I}$) should be produced by `\"I`, while the lowercase *ukranian i* ($\ddot{\imath}$) uses the command `\"\i`. Contrary to that example, the Khakassian letter *dotted-I*, which has a dot in uppercase form as well as in lowercase one, requires `\.I` to produce the uppercase variant $\dot{I}$, and an ordinary `i` to produce the lowercase variant $i$. Since it is assumed that *Mr. User* or *Mrs. User* can distinguish what letter does he/she want and to separate uppercase and lowercaee variants explicitly, it seems that the existance of two various types for specification of lowercase and uppercase forms does not produce any troubles. Unfortunately it is so before the commands `\uppercase` and `\lowercase` starts their work:

```
İ = \.I    \uppercase{\.I} = İ   \lowercase{\.I} = i
i =  i      \uppercase{i}  = I    \lowercase{i}  = i
Ï = \"I    \uppercase{\"I} = Ï   \lowercase{\"I} = ï
ï = \"\i   \uppercase{\"\i} = ï   \lowercase{\"\i} = ï
```

Although in principle `\uppercase` and `\lowercase` can be encountered at any place, in practice only the `\uppercase` and only in LATEX headers is used outside explicit User's control (although, for example, to create the headers using SMALL CAPITALS font one could insert in some style the macro containing the construction like `\lowercase{\sc ....}`). Roughly the LATEX mechanism for automatically created headers works as follows:

1. The header text *some-header-text* containing the text and TEX commands is created elsewhere.

2. The object *some-header-text* is expanded into non-expandable tokens.

3. The uppercase transformation over the list of non-expandable tokens is performed. The latter means that the characters are substituted by their `\uccode` values while the TEX primitive commands remain unchangable.

4. The result is substituted into the header of the page and the list of tokens is processed by TEX as usually.

(The detailed investigation of LATEX source code shows that it is not actually true, but the result of 'true' macro is close to this description.)

The behaviour of `\uppercase` and `\lowerrcase` is different in such formats as *Plain*, $\mathcal{AMS}$TEX and LATEX 2.09, and LATEX $2_\varepsilon$. The transformation of the characters in old formats uses only the values `\uccode` and `\lccode`, while LATEX $2_\varepsilon$ uses additional list `\@uclclist` of characters which use their own rule different from `\lccode`–`\uccode`. In particular, for old formats and LATEX$2_\varepsilon$ in OT1 with encoding the symbols `\i` and `\j` are not affected by the commands `\uppercase` and `\lowercase`, which means that the page header ctreated automatically by `\uppercase` will be wrong.

In this respect the behaviour of LATEX $2_\varepsilon$ in T1 encoding is quite different. The letter *dotless-i* remains constant after `\lowercase`, and is transformed into capital *I* after `\uppercase`. The letter *I* after `\uppercase` remais the same, but `\lowercase` transforms it into ordinary *i*. Finally, the letter *i* is transformed into *I* after `\uppercase`, and conserves its form after `\lowercase`. Similar transformations happen with $\jmath \longleftrightarrow J \longleftrightarrow j$. This property enables to create headers and footers properly as soon as only one `\uppercase` is used for it, but creates funny errors when the User tries to apply `\uppercase`–`\lowercase` by himself[6]. The optional variant where the stress is put above some of these letters makes the life even more unhappy.

The following macro enables to analyze if the text is transformed by some `\uppercase` or `\lowercase` commands and put correct form of the letters. The commands `\i` and `\I` create the letters $\imath/I$, which are correctly processed by `\uppercase`–`\lowercase`, commands `\ii` and `\II` create the letters $i/\dot{I}$ with corresponding properties:

```
\edef\temp{\the\i} \chardef\idotless=\temp\relax
\def\i{\ifnum`i=`\i \idotless \else I\fi}
\def\I{\ifnum`I=`\I I\else \idotless \fi}
\def\ii{\ifnum`i=`\i i\else \.I\fi}
\def\II{\ifnum`I=`\I \.I\else i\fi}
```

Finally, the ordinary letters `i` and `I` correspond to the case where the lowercase form has a dot while the uppercase form does not. These macro work due to

[6] The problem becomes even more complex taking into account that there is also the letter $\dot{I}$ in encoding T1. It is transferred to ordinary *i* by `\lowercase`, and into *I* after subsequent `\uppercase`.

the fact that the name of the macro `\I` and `\i` is conserved after `\uppercase`–`\lowercase` transformation, while in expressions `‘i` and `‘I` it is coverted appropriately. It is worth to note that although *then* and *else* parts of the macro are also affected by `\uppercase` and `\lowercase`, the results are correct:

```
\uppercase\expandafter{\i}=I   \uppercase\expandafter{\ii}=İ
\lowercase\expandafter{\i}=ı   \lowercase\expandafter{\ii}=i
\uppercase\expandafter{\I}=I   \uppercase\expandafter{\II}=İ
\lowercase\expandafter{\I}=ı   \lowercase\expandafter{\II}=i
```

It is more or less evident that the combination “lowercase-i-without-dot” + “uppercase-I-with-dot” is of no interest, although corresponding macro could be created as well.

Similar commands are intoduced for *ȷ/j* and *J/J̇*.

```
\edef\temp{\the\j} \chardef\jdotless=\temp\relax
\def\j{\ifnum‘j=‘\j \jdotless \else J\fi}
\def\J{\ifnum‘J=‘\J J\else \jdotless \fi}
\def\jj{\ifnum‘j=‘\j j\else \.J\fi}
\def\JJ{\ifnum‘J=‘\J \.J\else j\fi}
```

It is essential that the commands `\i` and `\I`, `\ii` and `\II`, `\j` and `\J`, `\jj` and `\JJ` creates the correct forms when the commands `\uppercase` and `\lowercase` or `\MakeUppercase` and `\MakeLowercase` affect them. They also work correctly when there are additional upper and lower accents created by the commands from the section 6. It is worth to note that the invisible characters created by the commands `\markchar` and `\MARKCHAR` described in section 7 are also transformed into each other after transformation by `\uppercase` and `\lowercase`.

### 10.1. Title letters ‘L+soft-sign’ and ‘N+soft-sign’

Similar problem of case-sensitivity exists in the writing systems for Serbian, Makedonian, Itelmen and Nganasan languages which use the ligatures ‘*L+soft-sign*’ and ‘*N+soft-sign*’: Лъ, лъ, Нъ, нъ. Except the uppercase and the lowercase variants shown above there is also the ‘title’ variant[7] which is composed from the uppercase Л and Н and the bowl from the *lowercase* soft-sign ъ. Such ligature is useful for titles where the first letter is the uppercase variant while the rest of the test is composed from the lowercase letters.

[7] The Dutch ligature *IJ* also have the title form.

The command `\uppercase` applied to the title letter should transfer it into the normal uppercase form, and `\lowercase` — into normal lowercase form. Changing `\lccode`–`\uccode` values for the title letters (which could help in this case) is not allowed in LaTeX $2_\varepsilon$, and the list `\@uclclist` used in LaTeX $2_\varepsilon$ to make lower/uppercase conversion is of no usage since different assignments are necessary for `\uppercase` and `\lowercase`.

The following macro uses the same idea to distinguish `\uppercase` and `\lowercase` modes and to produce correct letter as the output:

```
\def\makeCYRlje{%
   \ifnum`x=`\X        \CYRLJE  % there was \uppercase
   \else \ifnum `X=`\x \cyrlje  % there was \lowercase
   \else               \CYRlje  % natural title form
   \fi\fi
}
```

Here `\CYRLJE` corresponds to the uppercase form Лъ, `\cyrlje` — to the lowercase form лъ, and `\CYRlje` — to the title form. Similar commands can be defined for the letter 'N+soft-sign'.

## 11. Conditional operators if-then-else

Fine tuning of main characters and accents (modifiers) may require the analysis of the font style used for the composite construction. To make it easier, ACCENTBX contains special if-then-else macro:

`\ifupper{`*character*`}{`*if-upper*`}{`*if-lower*`}` — checks that you deal with the uppercase character: first parameter is the character to be checked, second — commands which are executed for *true*, third — commands to be executed for *false*.

`\iflower{`*character*`}{`*if-upper*`}{`*if-lower*`}` — checks that you deal with the lowercase character: first parameter is the character to be checked, second — commands which are executed for *true*, third — commands to be executed for *false*.

`\ifcaseupper{`*uppercase-or-none*`}{`*lowercase*`}` — checks that your construction was influenced by `\lowercase`: first group of commands is executed if there was `\uppercase` conversion or no conversion, and second group of commands is executed if there was `\lowercase` conversion.

`\ifcaselower{`*lowercase-or-none*`}{`*uppercase*`}` — checks that your construction was influenced by `\uppercase`: first group of commands is executed if there was `\lowercase` conversion or no conversion, and second group of commands is executed if there was `\uppercase` conversion.

`\ifcasetitle{`*no-case*`}{`*uppercase*`}{`*lowercase*`}` — checks allthree variants: first group of commands is executed if there is no conversion, second — for `\uppercase` conversion, third — for `\lowercase` conversion.

`\ifwide{`*character*`}{`*if-wide*`}{`*if-narrow*`}` — checks that you deal with wide character (character is considetred as wide if its width is more than the value `\accentwidefactor` (default value for it is 0.65em)).

`\ifwidth{`*character*`}{`*width*`}{`*if-wide*`}{`*if-narrow*`}` — just the same as `\ifwide`, but the width for comparison is specified explicitly as the second parameter.

if-then-else operators for font families: command `\ifrm{#1}{#2}` checks that you are in normal (upright roman) mode, command `\ifit{#1}{#2}` — that you are in *italic* mode, command `\ifsl{#1}{#2}` — that you are in *slanted* mode, command `\ifbf{#1}{#2}` — that you are in **boldface** mode, command `\ifsf{#1}{#2}` — that you are in sans serif mode, command `\ifsc{#1}{#2}` — that you are in SMALL CAPS mode, command `\iftt{#1}{#2}` — that you are in `typewriter` mode. There are also the commands `\ifmit{#1}{#2}` and `\ifsym{#1}{#2}` which checks that by some mistake you are in mathematical mode and are working with mathematical letters or mathematical symbols — but these should never used in practice 😀.

## 12. Changes in version 1.4

Version 1.4 is actually the first version which is uploaded to CTAN for wide distribution among TEX-users. Previous version 1.3 was distributed only between restricted set of people as a result of private communications. Nevertheless, the changes are the following:

— `\aboxadjust` is renamed into `\aboxshift`,

— `\makeaccent`, `\aboxframed` and `\aboxtuning` are added,

— commands `\akern` and `aboxmarker` appear,

— thickness and width od `\aboxbar` is controlled now by the User; macro `\aboxrule` appeared,

— optional arguments are introduced for some commands (by default, i.e., without such arguments, these commands behave as it was in previous version),

— robust versions of the commands with optional arguments are included,

— family of `\nullaccent...` commands is reviewed and modified (it was safe because in the previous version these commands were not documented and hence were not available for the User),

— behaviour of `\aboxbase` is changed — now it shifts the character only if it has positive depthy value; for compatibility with older application macro `\aboxbaseline` which shifts the character to its baseline always, is added,

— compatibility with *Plain*, $\mathcal{AMS}$TEX, etc., is checked and some errors are corrected,

— internal macro `\slant@value` is renamed and macro `\strip@pt` is deleted (since LATEX $2_\varepsilon$ also has such macro),

— if-then-else structures now have two or three parameters — groups of commands to be executed when corresponding condition is satisfied are specified explicitly,

— if-then-else structures for uppercase/lowercase conversion and for title letters are added,

— Now you have more or less detailed documentation ☺.

## Acknowledgements

I am grateful to Olga Lapko, Mikhail Kolodin and Andrew Janishevskii for numerous useful discussions which result to the appearance of the style file ACCENTBX. The examples of TEX programming demonstrated by Kees van der Laan helped to make the package ACCENTBX more effective and compact. The discussions with Jörg Knappen helped to understand better the base ideology of accents in T1 and EC/DC fonts. The extensive comments and recommendations of the first referee of this paper, Bernd Raichle, enabled to improve greatly the description of the `\accent` algorithm in section 2 and to correct numerous errors in suggested macro. Robin Fairbairns spent a lot of his time polishing the text of the first variant of this paper. I express my warmest thanks to all these TEXnicians and TEXperts.

And I would like to thank separately Aaldert Compagner from Delft University of Technology for his long-term friendship and patient attention as a teacher — not necessarily relating to some TEX or LATEX joint works.

## Bibliography

[1] Jörg Knappen, documentation supplementary to European Computer Modern font family.

[2] Donald E. Knuth. The TEXbook. Addison-Wesley, 1984 (reprinted with corrections in 1989).

[3] Victor Eijkhout. TEX by Topic, a TEXnician's Reference. Addison-Wesley, 1991.

[4] D.Salomon. NTG's Advanced TEX course: Insights & Hindsights. MAPS special issue, 1994.

[5] Eitan M. Gurari. TEX & LATEX: drawing & literate programming. McGraw-Hill, Inc., 1994.

[6] Kresten Krab Thorup and Frank Jensen, The style file CALC.STY (infix notation arithmetic in LATEX, version 4.0c).

[7] Michael Mehlich, The package *FP.sty* (*Fixed Point Package*), version 0.8 (April 1995).

[8] Frank Buchholz, The style file REALCALC.STY, dated January 1993.

# TEXmerge A System For Generating Form-based Documents

William M. Richter

*Texas Life Insurance*
*Waco, Texas, USA*
*hcswmr@texlife.com*

## 1. Introduction

TEXmerge is a C-language API for merging variable data into a pre-existing TEX document. The API is simple, light-weight, and easy to integrate into applications. The power and flexibility of TEX make the API useful in high-capacity production-oriented systems.

## 2. The TEXmerge API

The API consists of a small number of functions, here listed in the normal order of use. Most functions return an integer result code. Zero implies successful return. The integer result may be passed to the function `TeXmerge_GetErrorString()` to retrieve the corresponding error text string.

### 2.1. `TeXmergeName_t *TeXmerge_AllocNames(int count)`

Variable data is passed to TEXmerge as an associative array. Each entry in the array is a name/value pair. This function allocates an array of such name/value pairs containing `count` entries.

### 2.2. `TeXmergeName_t *TeXmerge_FreeNames(TeXmergeName_t *array, int count)`

This functions frees a name/value array. Obviously this one is not listed in order of use!

**2.3.** `int TeXmerge_SetArrayEntry(char *name, char *value, TeXmergeName_t *entry)`

This function sets `entry` to the passed `name` and `value` strings. The next two functions are for convenience and their use is not mandatory. In this and the next function, `value` may be passed as NULL to indicate 'no value'.

**2.4.** `int TeXmerge_SetName(char *name, char *value, TeXmergeName_t *array, int count)`

This function searches `array` for an entry whose name value matches `name`, and then sets the entrie's value to `value`.

**2.5.** `char *TeXmerge_GetName(char *name, TeXmergeName_t *array, int count)`

This function searches `array` for an entry whose name value matches `name`, and then returns the entries value string. The function returns NULL is `name` is not found in `array`.

**2.6.** `char *TeXmerge_GetNames(char *pathname, TeXmergeName_t **array, int *count)`

This function searches the `.tex` file specified in `pathname` for occurrences of lines having the format:

```
\texmergevar NAME
```

Lines of this form enumerate the merge variables that the document will use. The function returns a list of the names in the pointer variable pointed to by `array`. The number of elements in the array is returned in the integer pointed to by `count`. `array` should be freed when no longer needed with a call to `TeXmerge_FreeNames()`.

**2.7.** `int TeXmerge_OpenOutput(char *pathname, FILE **outFP, char *preamble)`

This functions creates an output file named `pathname`. An opened `FILE` pointer is returned in `outFP`. `preamble` is an optional "snippet" of TEX code that should be written at the beginning of the file. If `preamble` is passed as null, then no preable code is generated.

**2.8.** `int TeXmerge(char *pathname, TeXmergeName_t *array, int count, FILE *fp, int options)`

This function is the heart of the API. `pathname` is the name of a TEX form file containing invocations of macros whose names are the name values set in the passed `array`. `fp` is the `FILE` pointer returned by `TeXmerge_OpenOutput()`. `options` controls the merge operation. Currently the only option is whether or not to draw a frame around the merged variables (TXM_FRAMEVARS).

**2.9.** `int TeXmerge_CloseOutput(FILE *fp)`

After all invocations of the above functions are complete, this function should be called to close the output file. `fp` is the `FILE` pointer returned from `TeXmerge_OpenOutput()`.

**2.10.** `int TeXmerge_Process(char *pathname, char *dvidrv_name)`

Once the output file has been closed, it is ready for backend processing by TEX. This function invokes TEX and then the dvi driver named in `dvidrv_name`. All temporary `.log` and `.dvi` files are removed after use.

**2.11.** `int TeXmerge_View(char *pathname, int waitOption)`

A convenience function to run TEX and then run `xdvi`. `waitOption` is one of `TXM_WAIT` or `TXM_NOWAIT`. If `TXM_NOWAIT` is passed, then the current process if forked and then `xdvi` is run in a child process.

3. `int TeXmerge_Print(char *pathname, char *lpargs, char *output_pathname)`

A convenience function to run TEX and then run `dvilj`. If `lpargs` is non-NULL then it is used as switches for the `lp` command and the resulting `.lj` file will be queued for printing via the `lp` system. The pathname of the resulting `.lj` is returned in the character array pointed to by `output_pathname`.

**3.1.** `char *TeXmerge_GetErrorString(int)`

This functions returns a character string description corresponding to the passed integer value.

# 4. Example Program

A straightforward application of the TEXmerge API is illustrated in the following simple C code. This program,

— Creates an associative array with three elements.

— Sets the elements to have names `THISVAR`, `THATVAR`, and `ANOTHERVAR`, respectively.

— Opens an output file.

— Merges the associative array into an existing TEXfile called `test_form.tex` to create a temporary file called `temp.tex`.

— Closes the output file and processes it for viewing with `xdvi`.

```
#include "stdio.h"
#include "TeXmerge.h"

int main(int argc, char **argv) {
TeXmergeName_t *array?
int count=3?
int ret?
FILE *fp?
char *out_pathname="temp.tex"?
char *form_pathname="test_form.tex"?

   array = TeXmerge_AllocNames(count)?
   TeXmerge_SetArrayEntry("THISVAR", "some value", &array[0])?
   TeXmerge_SetArrayEntry("THATVAR", "blah, blah", &array[1])?
   TeXmerge_SetArrayEntry("ANOTHERVAR", "la-te-dah", &array[2])?
   ret = TeXmerge_OpenOutput(out_pathname, &fp, 0)?
   if (ret != TXM_OK) {
      fprintf(stderr, "TeXmerge_OpenOutput(%s): %s\n", out_pathname,
TeXmerge_GetErrorString(ret))?
      return(1)?
   }
   ret = TeXmerge(form_pathname, array, count, 0)?
   if (ret != TXM_OK) {
      fprintf(stderr, "TeXmerge(%s): %s\n", form_pathname,
TeXmerge_GetErrorString(ret))?
      return(1)?
```

```
}
TeXmerge_CloseOutput(fp)?
TeXmerge_View(out_pathname, TXM_WAIT)?
return(0)?
```

After the call to `TeXmerge_CloseOutput()` the contents of `temp.tex` would appear as follows:

```
\def\THISVAR{some value}
\def\THATVAR{blah, blah}
\def\ANOTHERVAR{la-te-dah}
\input test_form.tex
\bye
```

`test_form.tex` can have any TEX code of your choosing, including invocations of `\THISVAR`, `\THATVAR`, and `\ANOTHERVAR`.

## 5. Python Binding

A Python binding for the TEXmerge API is also available. A reimplementation of the previous C-code is given below.

```
import TeXmerge
import sys

array = {'THISVAR': 'some value',
            'THATVAR': 'blah, blah',
            'ANOTHERVAR': 'la-te-dah'}
out_pathname = 'temp.tex'
form_pathname = 'test_form.tex'
try: fp = TeXmerge.openOutput(out_pathname)
except IOError, errmsg:
sys.stderr.write('TeXmerge.openOutput(%s): %s\n' %
        (out_pathname, errmsg))
TeXmerge.merge(form_pathname, array, fp)
TeXmerge.closeOutput(fp)
TeXmerge.view(out_pathname, TeXmerge.TXM_NOWAIT)
```

Notes:

— The Python version is much more clean,

— Error detection via return values has been replaced with Python's exception mechanism. i.e. Instead of methods returning integer result codes they throw exceptions of the appropriate type which may be caught via the `try/except` construct,

— The `TeXmergeName_t` arrays used in the C-code example just use simple Python dictionary objects. As a result the nagging `count` integer with tracks the number of array elements is no longer needed.

— Constants defined in `TeXmerge.h` are accessed as attributes of the Python `TEXmerge` module.

# 6. Applications of TEXmerge at Texas Life

The TEXmerge API has been used to build a number of independent applications.

## 6.1. Interactive TEXmerge

The initial application of the API was in an X-windows program called TEXmerge. In this application a given directory holds a number of TEX-based form letters. The application allows the user to select a document, which is then scanned for merge variable names (via a call to `TeXmerge_GetNames()`). A frame is then displayed containing prompts and text-entry fields for each variable name. Once data is entered in all the text fields, the print function initiates the merge/print process, producing a completed document which may be viewed on-screen or printed.

At Texas Life the application was integrated with the policy administrative database in the following fashion. A policy database interface layer was built to return a associative array of selected basic data given a policy number. If one of the merge variables in the form document is named `POLNUM`, then when a value is entered in that field the policy database interface is used to make a query using the entered `POLNUM` value. If a match is found in the database, then other merge variable fields on the frame that have matching names in the policy data array are automatically populated with the corresponding retrieved data.

## 6.2. Policy Print

Each policy issued by Texas Life requires a *policy contract* to be printed. Most contracts are around twenty pages in length and include a large amount of variable data depending on the insurance product, the insured's issue age, smoker class, and other factors. Contracts also vary by the state in which they are written. Variable data is prepared by the policy administrative system, and the TEXmerge API is used to merge the data into the appropriate contract form to produce a printable document. Because decisions may be made directly in the TEX document (via `\ifx` and others) a single `.tex` file may be used to generate policy forms for multiple states.

## 6.3. Automated Correspondence

Texas Life's policy administrative system generates a large amount of correspondence to policyholders. These letters are form letters written in TEX, and the TEXmerge API is used to merge the data produced by the admin. system onto the appropriate letter form.

## 6.4. Customized Applications

Various other administrative applications that perform complex tasks have integrated the TEXmerge API as the facility to produce specialized printed output.

## 6.5. Forms

Texas Life uses a number of pre-printed forms that must be filled out the policy holders, agents, internal employees, etc, to request specific tasks to be performed (i.e., change of address, change of beneficiary, etc) These forms are produced using TEX. The resulting forms are used in two different ways. One is in the interactive TEXmerge application where the form may simply be printed or filled out online. The second is for selected forms, PostScript and PDF output is placed on the Texas Life web site for downloading by appropriate users.

Forms-on-the-web is currently an active area a development. A work-flow system is being built that will turn simple downloading of forms into a transaction management system. Instead of policyholders or agents downloading forms, filling them out, signing them and returning them to Texas Life, they will initiate a transaction via their web browser and enter data online that they would normally write on a pre-printed form. After the data is entered a

"turn-around" key will be assigned to the transaction and the form will be will printed populated with the data entered and a bar-coded turn-around key. The policyholder/agent mails the form to Texas Life (most forms require handwritten signatures) where it is scanned into the document management system and the specific transaction resumes processing via the turn-around key detected on the document.

### 6.6. Document Archival

Documents produced by all of the above systems are archived in a centralized repository call the document imaging system. Storing TEXmerge documents is efficient? only two items are stored to reproduce a document. One is a pointer to the `.tex` form document (there is a way to accommodate historical revisions of any specific document). The second is the associative data array for the variable data in the document. Each time the document is displayed, a sort of "JIT" technology is used. TEXmerge first merges the variable data into the form, then TEX is run on the merged form, and finally, `xdvi` or `dvilj` is invoked to display/print the resulting document.

# TEXνικές: Σελίδες με φόντο

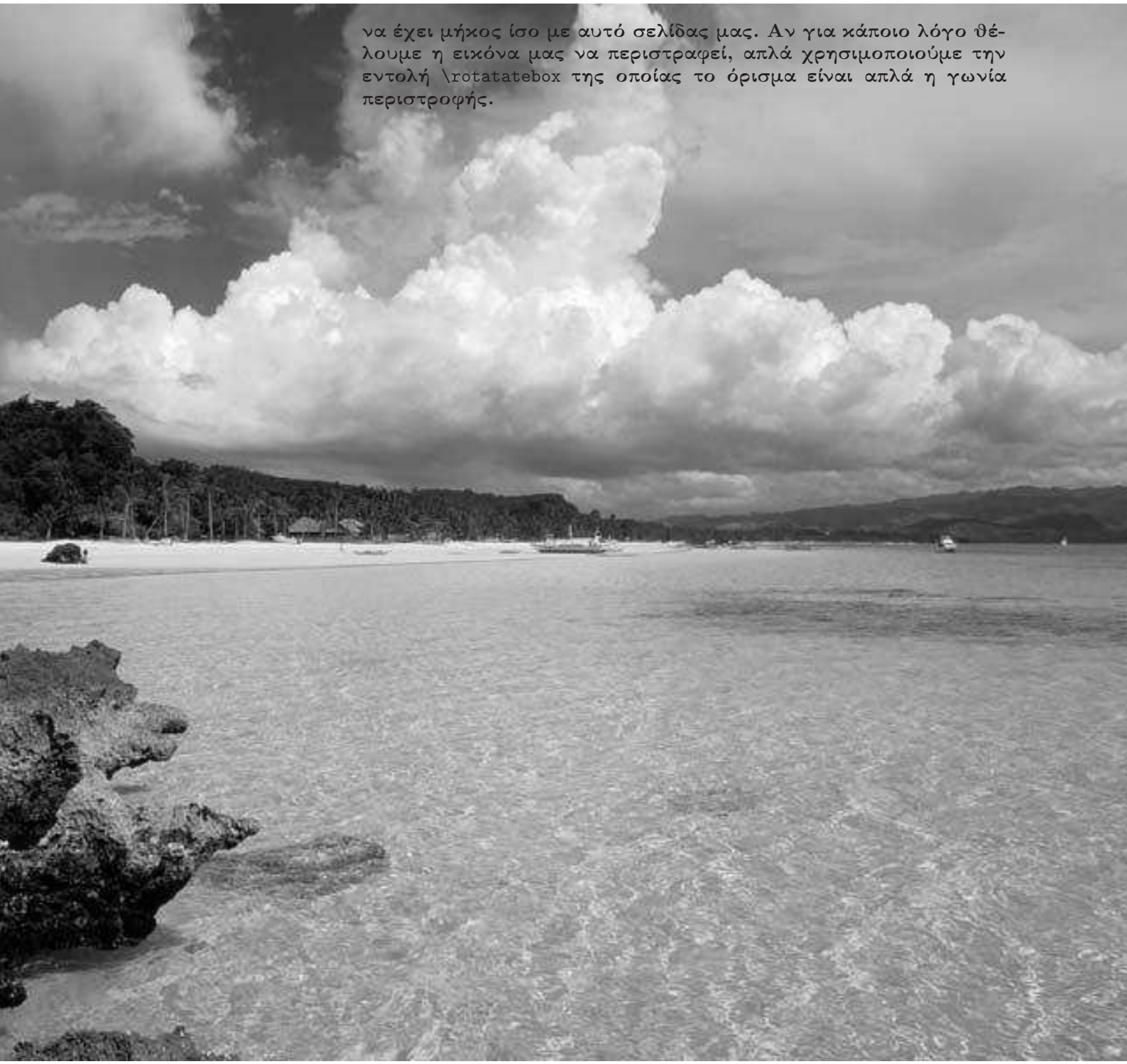
Απόστολος Συρόπουλος

*28ης Οκτωβρίου 366*
*671 00 Ξάνθη*
*E-mail:* `apostolo@obelix.ee.duth.gr`

Εάν υποθέσουμε ότι θέλουμε να δημιουργήσουμε ένα διαφημιστικό φυλλάδιο χρησιμοποιώντας το LATEX θα πρέπει να είμαστε σε θέση να βάλουμε μια ολόκληρη εικόνα σε φόντο, ακριβώς όπως αυτό που βλέπετε στην παρούσα σελίδα. Το ερώτημα φυσικά είναι: πώς μπορεί να γίνει αυτό; Καταρχήν πρέπει να πούμε ότι η εικόνα μας θα πρέπει να είναι σε μορφή Encapsulated PostScript. Αν η εικόνα μας είναι σε οποιαδήποτε άλλη μορφή, τότε απλά πρέπει να την μετατρέψουμε σε μορφή Encapsulated PostScript με κάποιο πρόγραμμα όπως το jpeg2ps (βλ. Σ. Κοντογιάννη, «jpeg2ps: μετατροπή αρχείων JPEG σε PostScript», *Εὔτυπον*, νο. 4, σελ. 71–73). Έχοντας την εικόνα στην κατάλληλη μορφή, θα πρέπει να χρησιμοποιείσουμε δύο πακέτα τα οποία μας επιτρέπουν να βάλουμε μια τέτοια εικόνα ως φόντο σε μία σελίδα. Το πρώτο πακέτο είναι το graphicx, το οποίο κάνει δυνατή την ενσωμμάτωση γραφικών σε αρχεία LATEX. Το δεύτερο πακέτο είναι το eso-pic το οποίο είναι ουσιαστικά το πακέτο που επιτρέπει να βάλουμε την εικόνα σε φόντο. Παρακάτω δίνουμε τον κώδικα που πρέπει κανείς να χρησιμοποιήσει ώστε να βάλει φόντο σε μία σελίδα A4 την εικόνα pict290.eps.

```
\AddToShipoutPicture{%
   \setlength{\unitlength}{1mm}
 \put(0,0){\makebox(210,296)[t]{%{\rotatebox{180}{%
 \includegraphics[height=296mm]{pict290.eps}}}}%}
```

Η εντολή \setlength καθορίζει ποιο θα είναι το προκαθορισμένο μήκος όταν αναφερόμαστε σε διαστάσεις, στην περίπτωση μας αυτό είναι το 1 mm. Ακριβώς μετά βάζουμε στην θέση (0,0) της σελίδας μας ένα *κουτί* με διαστάσεις ίσες προς αυτές τις σελίδας μας. Στο κουτί βάζουμε την εικόνα η οποία πρέπει



**να έχει μήκος ίσο με αυτό σελίδας μας. Αν για κάποιο λόγο θέλουμε η εικόνα μας να περιστραφεί, απλά χρησιμοποιούμε την εντολή** `\rotatatebox` **της οποίας το όρισμα είναι απλά η γωνία περιστροφής.**

# Βιβλίο-Παρουσίαση

Δημήτριος Ἀ. Φιλίππου

*Κάτω Γατζέα*
*385 00 Βόλος*

**Δυὸ λόγια γιὰ τὴν Βιβλίο-Παρουσίαση** — Ἑτοῦτες οἱ σελίδες σκοπὸ ἔχουν νὰ παρουσιάσουν ὁρισμένα βιβλία ποὺ θὰ ἐνδιέφεραν τὶς TEXνίτριες καὶ τοὺς TEXνίτες. Κάθε συνδρομὴ στὴν παρουσίαση νέων βιβλίων (ἀλλὰ καὶ ὄχι τόσο νέων) γύρω ἀπὸ τὸ TEX καὶ τὴν τυπογραφία εἶναι καλοδεχούμενη. (Οἱ περίεργοι ἀναγνῶστες μποροῦν ἐπίσης νὰ σημειώσουν ὅτι τὸν τίτλο «Βιβλίο-Παρουσίαση» τὸν δανειστήκαμε ἀπὸ κάποιες καθημερινὲς ἐφημερίδες, ἀλλὰ ἐλπίζουμε ὅτι — μιᾶς καὶ πρόκειται γιὰ ἕνα τόσο μικρὸ δάνειο — θὰ μᾶς τὸ χαρίσουν.)

***

**Micheal Doob,** `T pr to b¤ma st` ***TEX***, Ἐκδόσεις Παρατηρητής, Θεσσαλονίκη 2000 (μτφ. Δημήτριος Ἀ. Φιλίππου). Σελ. viii + 143. ISBN 960-374-081-0. Τιμὴ 3.120 δρχ. (9 €). Διατίθεται ἀπὸ ὅλα τὰ βιβλιοπωλεῖα.

Λένε ὅτι ἐὰν δὲν παινέψεις τὸ σπίτι σου, τότε αὐτὸ θὰ πέσει νὰ σὲ πλακώσει. Ὅμως, ὁ συνήθης ἔνοχος ἐτούτης τῆς στήλης θέλει ἁπλῶς νὰ σᾶς πεῖ ὅτι μετέφρασε καὶ κυκλοφόρησε στὰ ἑλληνικὰ ἕνα ἀκόμα βιβλίο γιὰ τὸ TEX. Τὸ βιβλίο τοῦ Michael Doob (Μάικλ Ντούμπ), ποὺ εἶναι γνωστὸ ἀπὸ προηγούμενη ἔκδοση ποὺ κυκλοφόρησε δωρεὰν μέσῳ τοῦ Διαδικτύου, ἀπευθύνεται στὸν πρωτάρη τοῦ TEX καὶ ὄχι στὸν πρωτάρη τοῦ LATEX. Ὡστόσο, ἀκόμα καὶ ἐκεῖνοι ποὺ ἀσχολοῦνται ἀποκλειστικὰ μὲ τὸ LATEX θὰ βγοῦν κερδισμένοι διαβάζοντας ἕνα τόσο μικρὸ βιβλίο ποὺ εἶναι ἰδιαιτέρως προσιτὸ στὴν τιμή του. Ἄλλα λόγια δὲν προσθέτουμε, παρὰ μόνο ὅτι κάθε κριτικὴ εἶναι ε ὐ π ρ ό ς δ ε κ τ η καὶ θὰ τὴν δημοσιεύσουμε εὐχαρίστως.

***

**John H. Bowman,** ***Greek Printing Types in Britain – From the Late Eighteenth to the Early Tewntieth Century***, Typophilia (Τυποφιλία), Θεσσαλονίκη 1998. Στὰ ἀγγλικά. Σελ. 342. ISBN 960-7285-20-4. Τιμὴ 10.000 δρχ. (30 €). Διατίθεται ἀπὸ τὸ βιβλιοπωλεῖο «Τυποφιλία», Δελλίου 6, 546 21 Θεσσαλονίκη, τηλ. 031-239823, φὰξ 031-269651.

Ὁ Κλήμης Μαστορίδης, ἡ ψυχὴ τῶν θεσσαλονικιώτικων Ἐκδόσεων Τυποφιλία, διαλέγει προσεκτικὰ τὰ βιβλία ποὺ ἐκδίδει. Τέσσερα χρόνια μετὰ τὴν ἔκδοση τοῦ λευκώματος τοῦ Victor Scholderer, στὸ ὁποῖο ἀναφερθήκαμε στὸ προηγούμενο τεῦχος, ὁ Κλήμης Μαστορίδης «ξαναχτύπησε» μὲ ἕνα βιβλίο μοναδικὸ στὸ εἶδος του: τὴν διδακτορικὴ διατριβὴ τοῦ John H. Bowman (Τζὸν Χ. Μπόουμαν) μὲ θέμα τὰ ἑλληνικὰ τυπογραφικὰ στοιχεῖα στὴν Βρετανία ἀπὸ τὸ τέλος τοῦ 18ου ἕως τὶς ἀρχὲς τοῦ 20οῦ αἰώνα.

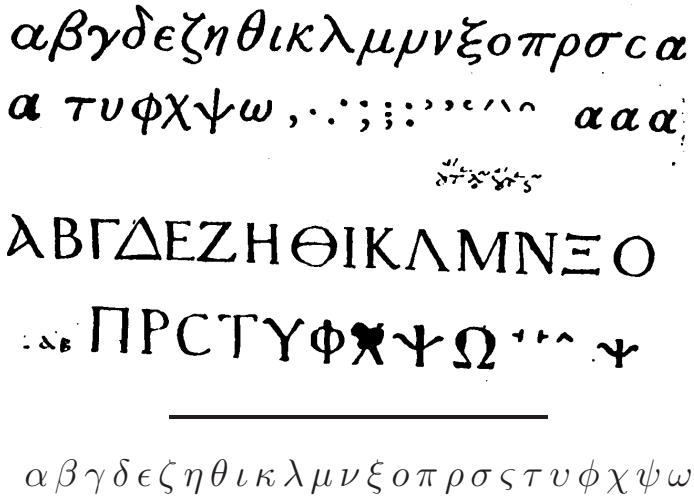
Ἡ Βρετανία, χώρα μὲ μεγάλη παράδοση στὴν Τυπογραφία, ὑπῆρξε ὁ τόπος ὅπου σχεδιάσθηκαν μερικὲς ἀπὸ τὶς πιὸ ὄμορφες γραμματοσειρὲς γιὰ ὅλα τὰ ἀλφάβητα. Βρετανοὶ σχεδιαστὲς στοιχείων, ὅπως ὁ John Baskerville (Τζὸν Μπάσκερβιλ, 1707–1775) καὶ ὁ Eric Gill (Ἔρικ Γκίλ, 1882–1940), καθὼς καὶ βρετανικὲς ἑταιρεῖες παραγωγῆς τυπογραφικῶν στοιχείων, ὅπως ἡ Monotype, ἐπηρέασαν καὶ ἐξακολουθοῦν νὰ ἐπηρεάζουν βαθιὰ τὴν τυπογραφική μας αἰσθητική.

Ἡ ἐπαφὴ τῶν Βρετανῶν μὲ τὰ ἔργα τῶν ἑλλήνων συγγραφέων τῆς κλασικῆς ἀρχαιότητας δημιούργησε τὴν ἀνάγκη σχεδιασμοῦ ἑλληνικῶν στοιχείων. Τὰ στοιχεῖα αὐτὰ χρησιμοποιήθηκαν κυρίως γιὰ τὶς ἐκδόσεις κλασικῶν συγγραμάτων ἀπὸ τὰ βρετανικὰ πανεπιστήμια ἀπὸ τὸν 17ο αἰ. ἕως σήμερα. Μάλιστα, ὁ Richard Porson (Ρίτσαρντ Πόρσον, 1759–1808), καλλιγράφος καὶ καθηγητὴς τῶν Ἀρχαίων Ἑλληνικῶν στὸ Πανεπιστήμιο τοῦ Καίμπριτζ, ἐπηρέασε πάρα πολὺ τὴν ἀγγλοσαξωνικὴ ἀντίληψη γιὰ τὰ ἑλληνικὰ τυπογραφικὰ στοιχεῖα. Πολλὰ βιβλία ἀρχαίων ἑλλήνων συγγραφέων ἐξακολουθοῦν νὰ τυπώνονται καὶ σήμερα μὲ πλάγια ἑλληνικὰ στοιχεῖα τύπου Porson. Ακόμα καὶ ὁ Donald Knuth χρησιμοποίησε (ἴσως ἐμμέσως) ὡς πρότυπο τοὺς χαρακτῆρες τοῦ Porson γιὰ τὸν σχεδιασμὸ τῶν ἑλληνικῶν μαθηματικῶν συμβόλων τοῦ TEX (Εἰκόνα 1).

Θέλετε λοιπὸν νὰ ἐμβαθύνετε στὶς ἀπαρχὲς τῆς ἑλληνικῆς Τυπογραφίας; Θέλετε νὰ μάθετε γιατί εἶναι λάθος νὰ μιλᾶμε γιὰ περισπώμενα βραχέα φωνήεντα *ε*̃ καὶ *ο*̃, ἔστω κι ἂν ἰσχυρίζονται τὸ ἀντίθετο οἱ δημιουργοὶ τοῦ *Ὦμέγα*; Θέλετε νὰ δεῖτε τὶς λεπτομερειακὲς σχεδιαστικὲς διαφορὲς τοῦ *ζ*, τοῦ *π* καὶ τοῦ *ρ* ἀπὸ γραμματοσειρὰ σὲ γραμματοσειρά; Θέλετε νὰ δεῖτε τὰ πραγματικὰ ἑλληνικὰ στοιχεῖα Baskerville, ποὺ δὲν ἔχουν καμία σχέση μὲ τὰ ὅσα Baskerville Greek κυκλοφοροῦν σήμερα; Θέλετε μήπως νὰ σχεδιάσετε μία νέα ἑλληνικὴ γραμματοσειρά; Δὲν ἔχετε παρὰ νὰ συμβουλευθεῖτε τὸ βιβλίο τοῦ Bowman. Πρόκειται γιὰ ἕνα βιβλίο πολὺ εὐχάριστο στὴν ἀνάγνωση ποὺ ξεδιαλύνει πολλὲς ἀπορίες.

Ἂς ἐλπίσουμε ὅτι κάποια ἡμέρα, κάποιοι ἄλλοι θὰ ἀκολουθήσουν τὸ παράδειγμα τοῦ Bowman γιὰ νὰ καταγράψουν ἐπίσης τὴν λεπτομερὴ ἱστορία τῶν ἑλληνικῶν τυπογραφικῶν στοιχείων ποὺ σχεδίασαν ἰταλοὶ καὶ γάλλοι τυπογράφοι, δύο ἄλλες τυπογραφικὲς σχολὲς μὲ ἐξ ἴσου σημαντικὴ ἐπίδραση (ἴσως καὶ περισσότερη) στὴν ἑλληνικὴ Τυπογραφία.

*
* *
*




**Εἰκόνα 1:** Ἐπάνω χειρόγραφο ἑλληνικὸ ἀλφάβητο τοῦ Richard Rorson (ἀπὸ τὸ βιβλίο τοῦ John. H. Bowman, *Greek Printing Types in Britain*, Τυποφιλία, Θεσσαλονίκη 1998, σελ. 108) καὶ κάτω τὰ ἑλληνικὰ μαθηματικὰ σύμβολα τοῦ Donald Knuth. Προσέξτε τὶς ὁμοιότητες· ἡ μόνη οὐσιαστικὴ διαφορὰ βρίσκεται στὸ $\nu$, ποὺ ὁ Porson, ἀκολουθώντας τὴν βυζαντινὴ παράδοση, τὸ ἔγραφε συχνὰ σὰν τὸ $\mu$.

**Chuck Musciano καὶ Bill Kennedy, *HTML & XHTML: The Definitive Guide***, 4th ed., O'Reilly, Cambridge (Massachusetts, USA) 2000. Σελ. 652. ISBN 0-596-00026-X. Τιμὴ 34,95 δολ. ΗΠΑ (περίπου 13.000 δρχ. ἢ 38 €). Διατίθεται ἀπὸ τὰ βιβλιοπωλεῖα ξένων ἐπιστημονικῶν βιβλίων.[1]

Γιὰ ὅσους ἀναρωτιοῦνται τί εἶναι ἡ XHTML, ἡ ἀπάντηση εἶναι ὅτι πρόκειται γιὰ τὴν γλώσσα ἐπισήμανσης HTML (HyperText Markup Language) «ξαναγραμμένη» στὴν γλώσσα ἐπισήμανσης XML (eXtensible Markup Language). Οἱ εἰδήμονες τοῦ WEB μᾶς λένε ὅτι ἡ ὥρα νὰ μεταπηδήσουμε ἀπὸ τὴν HTML στὴν XML πλησιάζει ταχύτατα, καὶ ὅτι τὸ βιβλίο τῶν Musciano (Μουσιάνο) καὶ Kennedy (Κέννεντυ) μᾶς εἶναι ἀπολύτως ἀπαραίτητο. Ἐφ' ὅσον ὁ συνήθης ἔνοχος τῆς ΒΙΒΛΙΟ-ΠΑΡΟΥΣΙΑΣΗΣ κατέχει μόνο κάτι λίγα βασικὰ τῆς HTML, ἀφήνει κάποιον ἄλλο νὰ μιλήσει:

[1] Ὅσοι ἀναγνῶστες εἶστε μέλη τοῦ Συλλόγου *εφτ* μπορεῖτε νὰ παραγγείλετε τὰ βιβλία τῶν ἐκδόσεων O'Reilly ἀπ' εὐθείας ἀπὸ τὸν ἐκδοτικὸ οἶκο (`www.oreilly.com`) σὲ τιμὴ μειωμένη. Γιὰ περισσότερες πληροφορίες γιὰ τὸ πῶς θὰ κάνετε τὴν παραγγελία σας, ἀπευθυνθεῖτε στὸν Σύλλογο στὴν διεύθυνση: `eft@ocean1.ee.duth.gr`

«Ἔχω τὴν προηγούμενη ἔκδοση τοῦ βιβλίου, ὅπως καὶ τὸ *CGI Programming* ἀπὸ τὴν ἴδια σειρὰ τοῦ O'Reilly. Καὶ ὄντως, ὅσον ἀφορᾶ τὴν HTML, [τὸ βιβλίο τῶν Musciano καὶ Kennedy] εἶναι the "definitive guide".

»Τὸ βιβλίο, ὅπως καὶ τὰ ἄλλα τῆς σειρᾶς ξεφεύγουν ἀπὸ τὸ σύνηθες ὕφος ἄλλων manual καὶ guide. Δὲν θὰ βρεῖτε CD στὴν πλάτη τοῦ βιβλίου, οὔτε καραγκιοζάκια καὶ ἄλλους... assistant νὰ ξεπετάγονται ἀπὸ τὶς σελίδες τοῦ βιβλίου. Τὸ βιβλίο δὲν εἶναι ὅπως ἡ γνωστὴ σειρὰ τῆς SAMS "Teaching... in 24 hours" κ.τ.λ.

»Ἂν ὅμως θέλετε νὰ βρεῖτε μαζεμένα τὰ χαρακτηριστικὰ τῆς HTML μὲ σαφήνεια κι ἀκρίβεια καὶ παραδείγματα, τότε αὐτὸ τὸ βιβλίο εἶναι must.»

Ἰωάννης Δημάκος (`idimakos@otenet.gr`)
Μήνυμα στὴν λίστα `eft`, 16 Αὐγούστου 2000.

*
* *
*

**Norman Walsh καὶ Leonard Muellner, *DocBook: The Definitive Guide***, O'Reilly, Cambridge (Massachusetts, USA) 1999. Σελ. 652. Περιλαμβάνει CD-ROM. ISBN 1-56592-580-7. Τιμὴ 36,95 δολ. ΗΠΑ (περίπου 14.000 δρχ. ἢ 40 €). Διατίθεται ἀπὸ τὰ βιβλιοπωλεῖα ξένων ἐπιστημονικῶν βιβλίων.[2]

Ἐὰν εἴστε ἕτοιμοι νὰ μεταπηδήσετε ἀπὸ τὴν HTML στὴν XML, τότε ἴσως θὰ πρέπει νὰ σκεφθεῖτε νὰ χρησιμοποιήσετε τὸ DocBook DTD (Document Type Definition), ἕνα εἶδος προτύπου ἢ ἕνα σύνολο μακροεντολῶν ποὺ ἐπιτρέπει τὴν παραγωγὴ λογικὰ δομημένων κειμένων γιὰ παρουσίαση στὸ WEB, γιὰ ἐκτύπωση ἢ γιὰ ἀναπαραγωγὴ σὲ ἄλλα σύγχρονα ὀπτικοακουστικὰ μέσα. Γιὰ τοὺς ἀνίδεους, σὰν καὶ τὸν ἔνοχο ἑτούτης τῆς Βιβλίο-Παρουσίασης, ἡ σχέση τοῦ DocBook μὲ τὴν XML καὶ τὴν SGML μοιάζει κάπως μὲ τὴν σχέση τοῦ LaTeX μὲ τὸ TeX.

Ὅσοι θὰ μπεῖτε στὸν κόπο νὰ μπλέξετε μὲ τὸ DocBook, ἴσως στὴν ἀρχὴ νὰ χαθεῖτε μέσα στὴν νέα ὁρολογία τῆς XML/SGML. Θὰ πρέπει νὰ καταλάβετε τί εἶναι ἡ *δομημένη σήμανση* (structured, semantic markup), τὰ *στοιχεῖα σήμανσης* (elements), οἱ *ἰδιότητες* ἢ *γνωρίσματα* (attributes), οἱ *ὀντότητες* (entities) καὶ τὰ *φύλλα ὕφους* (style sheets). Ἐπιπλέον τὸ βιβλίο *DocBook: The Definitive Guide* ἀποτελεῖ ἕνα εἶδος καταλόγου γιὰ ὅλα τὰ στοιχεῖα σήμανσης τοῦ DocBook. Λείπει, γιὰ παράδειγμα, ἡ ευκολία στὴν ἀνάγνωση ποὺ χαρακτηρίζει τὸ ἐπίσημο ἐγχειρίδιο τοῦ Leslie Lamport γιὰ τὸ LaTeX. Κι ὅσοι ἐπιθυμεῖτε νὰ ἑτοιμάσετε κάτι στὰ ἑλληνικὰ ἴσως ἀπογοητευθεῖτε παντελῶς· τὸ DocBook, στὴν σημερινή του μορφή, ὑποστηρίζει μὲν ὁρισμένες δυτικοευρωπαϊκὲς γλῶσσες μὲ λατινικὸ

2 Βλ. ὑποσημείωση 1.

ἀλφάβητο, ἀλλὰ δὲν ὑποστηρίζει ἑλληνικά (§4.4.2). Τέλος, τὸ βιβλίο δὲν ἀναφέρει πουθενὰ τὰ ἐργαλεῖα ποὺ αὐτοματοποιοῦν τὴν σύνταξη καὶ παραγωγὴ κειμένων μὲ τὸ DocBook. Τὰ μόνα σχετικὰ ἐργαλεῖα ποὺ ὑπάρχουν αὐτὴ τὴν στιγμὴ εἶναι ἐμπορικά (π.χ., τὸ Epic τῆς Arbortext) καὶ κοστίζουν πανάκριβα (ἀπὸ 500 ἕως 25 000 €).

Ἴσως τὸ DocBook νὰ ἀποτελεῖ τὸ μέλλον (ἢ μάλλον μέρος τοῦ μέλλοντος) τῆς στοιχειοθεσίας λογικὰ δομημένων κειμένων. Ὡστόσο, γιὰ τὴν ὥρα, τὸ βασικὸ ἐγχειρίδιό του ἀπευθύνεται μόνο στοὺς πολὺ εἰδικούς. Ὅσοι ἔχετε τὸ κουράγιο νὰ μπεῖτε σὲ νέες περιπέτειες, μπορεῖτε νὰ βρεῖτε ὁλόκληρο τὸ βιβλίο *DocBook: The Definitive Guide* στὴν ἱστοσελίδα `http://www.ora.com/catalog/docbook/chapter/book/docbook.html`.

*
* *
*

**Christian Rolland, *LaTeX par la pratique***, O'Reilly, Paris 1999. Στὰ γαλλικά. Σελ. 580. Περιλαμβάνει τὸ CD-ROM *TeXLive4*. ISBN 2-84177-073-7. Τιμὴ 230 γαλλικὰ φράγκα (περίπου 12.000 δρχ. ἢ 35 €). Διατίθεται ἀπὸ τὰ βιβλιοπωλεῖα ξένων ἐπιστημονικῶν βιβλίων.[3]

Ἡ βιβλιογραφία τοῦ LaTeX σὲ γλῶσσες πέρα ἀπὸ τὰ ἀγγλικὰ εἶναι κάπως φτωχή. Στὸν κατάλογο τῆς διεθνοῦς βιβλιογραφίας γιὰ τὸ TeX καὶ τὸ LaTeX τοῦ γαλλικοῦ ἐργαστηρίου Πληροφορικῆς LORIA[4] ἀναφέρονται 46 σχετικὰ βιβλία στὰ ἀγγλικά, 17 στὰ γερμανικὰ καὶ 12 στὰ γαλλικά. Ὅλες οἱ ἄλλες γλῶσσες ἀκολουθοῦν ἀσθμαίνοντας μὲ ἕνα ἢ τὸ πολὺ δύο τίτλους.

Ὁ Christian Rolland (Κριστιὰν Ρολλάν) προσπαθεῖ νὰ καλύψει κάπως τὴν ἀπόσταση ποὺ χωρίζει τὴν γαλλικὴ ἀπὸ τὴν ἀγγλικὴ βιβλιογραφία τοῦ LaTeX. Μετὰ ἀπὸ τὸ πρῶτο του βιβλίο γιὰ τὸ LaTeX (*LaTeX, Guide pratique*, Addison-Wesley, α′ ἔκδ. 1993, β′ ἔκδ. 1995), ὁ Rolland ἐπιστρέφει μὲ ἕνα βιβλίο πολὺ πιὸ ἐκτενὲς καί — προφανῶς — πολὺ πιὸ ἐνημερωμένο.

Στὰ δεκαοκτὼ κεφάλαια καὶ στὶς σχεδὸν ἑξακόσιες σελίδες τοῦ βιβλίου, ὁ ἀναγνώστης εἰσάγεται στὸ LaTeX, ἀλλὰ καὶ ἐντρυφεῖ σ' αὐτό. Τὸ βιβλίο καλύπτει μὲ πολλὰ παραδείγματα ὅλα τὰ θέματα τῆς ἁπλῆς καὶ πολύπλοκης στοιχειοθεσίας μὲ τὸ LaTeX· τὸ κείμενο, τὰ μαθηματικά, τοὺς διάφορους καταλόγους (συμπεριλαμβανομένης καὶ τῆς βιβλιογραφίας) καὶ τοὺς πίνακες. Ὅμως ὁ Rolland ἐπεκτείνεται καὶ σὲ θέματα πέρα ἀπὸ τὸ καθ' αὐτὸ LaTeX, ὅπως γιὰ παράδειγμα, ἡ παρεμβολὴ

3 Βλ. ἐπίσης ὑποσημείωση 1.

4 Laboratoire Lorrain de Recherche en Informatique et ses Applications, βλ. `http://www.loria.fr/services/tex/english/texbib.html`.

γραφικῶν, ἡ διαχείριση μεγάλων ἀρχείων καὶ ἡ μετατροπὴ τῶν ἀρχείων LaTeX σὲ ἀρχεῖα HTML. Μὲ λίγα λόγια πρόκειται γιὰ ἕνα πλῆρες ἐγχειρίδιο τοῦ LaTeX.

*
* *
*

9 771108 417007 05>