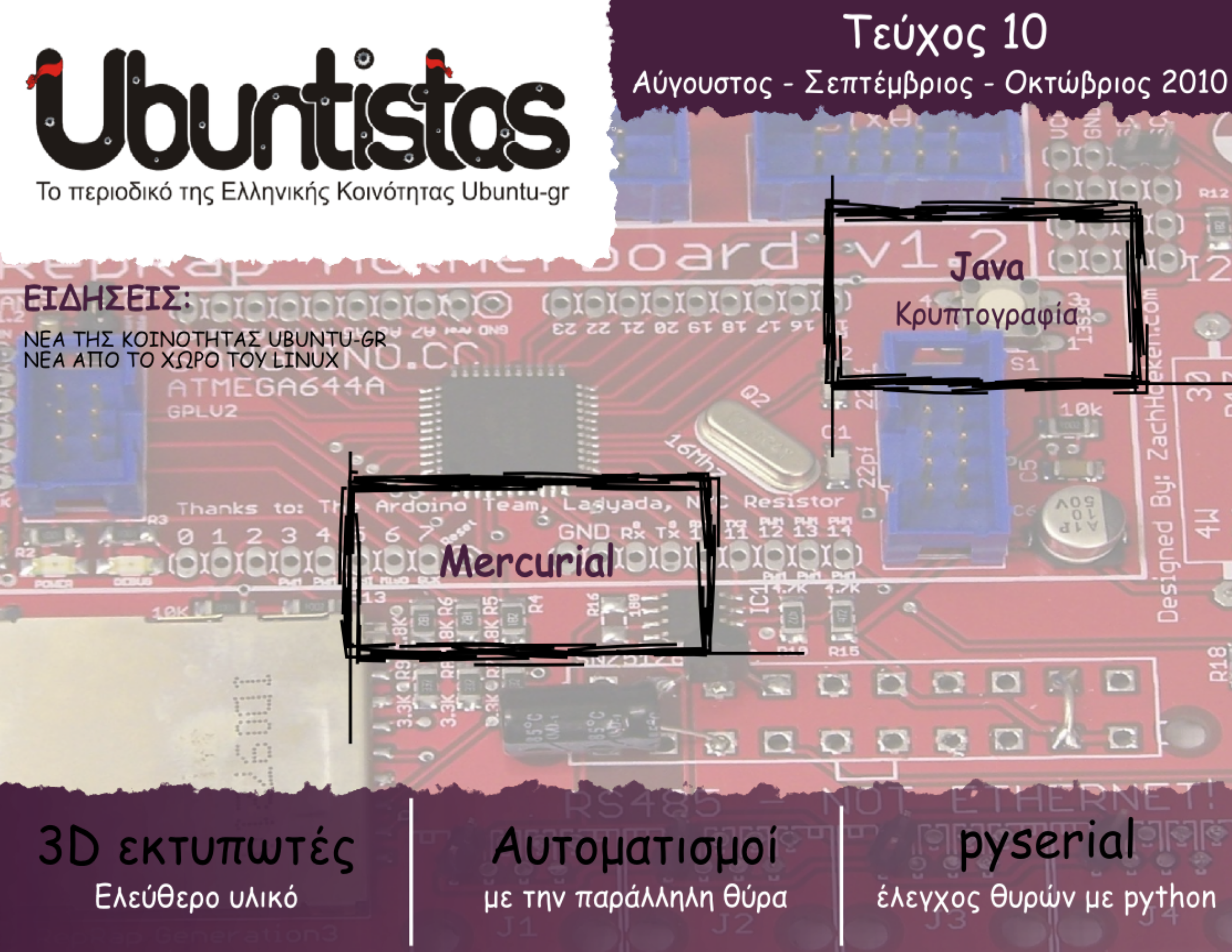
Ubuntistas

Το περιοδικό της Ελληνικής Κοινότητας Ubuntu-gr

Τεύχος 10
Αύγουστος - Σεπτέμβριος - Οκτώβριος 2010



ΕΙΔΗΣΕΙΣ:

ΝΕΑ ΤΗΣ ΚΟΙΝΟΤΗΤΑΣ UBUNTU-GR
ΝΕΑ ΑΠΟ ΤΟ ΧΩΡΟ ΤΟΥ LINUX

Java
Κρυπτογραφία

Mercurial

3D εκτυπωτές
Ελεύθερο υλικό

Αυτοματισμοί
με την παράλληλη θύρα

pyserial
έλεγχος θυρών με python

Ubuntistas
Το περιοδικό της Ελληνικής Κοινότητας Ubuntu-gr

Τεύχος 10 - Αύγουστος Σεπτέμβριος Οκτώριος 2010

Ομάδα Περιοδικού:

- **Κωστάρας Γιάννης (hawk):**
  Συντάκτης - jkost@freemail.gr
- **Παπαδόπουλος Δημήτρης (Dimitris):**
  Συντάκτης, σελιδοποίηση - chaosdynamics@googlemail.com
- **Πετούμενου Τζένη (jennie):**
  Επιμελήτρια Κειμένων - epetoumenou@gmail.com
- **Πούλιος Κωνσταντίνος (logari81):**
  Συντάκτης - poulios.konstantinos@googlemail.com
- **Χατζηπαντελής Παντελής (kalakouentin):**
  Συντάκτης - kalakouentin@yahoo.com
- **(the_eye):**
  Συντάκτης
- **Δήμος Πούπος (Qdata):**
  Επιμελητής - demos_w57@hotmail.com

Σημείωμα από τη σύνταξη...

Ένα ακόμη καθυστερημένο τεύχος του ubuntistas έρχεται στην επιφάνεια. Δε θα αναλωθώ σε δικαιολογίες, γιατί υπάρχουν πολλές, αλλά θα επικεντρωθώ στο γεγονός ότι είμαστε λίγοι μεν αλλά ενεργοί.

Το παρόν τεύχος διαπραγματεύεται μια διαφορετική πτυχή του ΕΛΛΑΚ: το ελεύθερο υλικό. Πόσο διαφορετική θα ήταν η ζωή μας αν τα σχέδια για το ψυγείο, το φούρνο και το πλυντήριο ήταν ελεύθερα; Δε θα γεννιούνταν νέες ιδέες πιο εύκολα; Δε θα βελτιώνονταν τα υπάρχοντα προϊόντα πιο γρήγορα; Τι θα γινόταν αν τα σχέδια για τα έπιπλα, τα σκεύη, και οτιδήποτε άλλο, ήταν ελεύθερα για όποιον ήθελε να τα κατασκευάσει; Δε θα είχαν δημιουργηθεί νέες τάσεις και ίσως και νέες τεχνολογίες; Το άρθρο για τον 3D εκτυπωτή εξερευνά νέες, πραγματικά ενδιαφέρουσες προοπτικές. Το αφιέρωμα στο ελεύθερο υλικό συνεχίζεται με άρθρα για τον έλεγχο της παράλληλης και της σειριακής θύρας του υπολογιστή με διάφορες γλώσσες προγραμματισμού, όπως C και python.

Το τεύχος βέβαια περιλαμβάνει και τυπικά άρθρα λογισμικού, όπως το δεύτερο μέρος του άρθρου για την κρυπτογραφία στη Java, καθώς και την παρουσίαση του mercurial, ενός συστήματος ελέγχου εκδόσεων, απαραίτητου για τη συστηματική διαχείριση του κώδικα.

Αυτά με το, ισχνό μεν, αλλά ενημερωτικό τεύχος μας.

Καλή ανάγνωση!

Περιεχόμενα

Το ubuntu

Το ubuntu linux είναι ένα λειτουργικό σύστημα. Με περιβάλλον εργασίας gnome το φωνάζουμε ubuntu, με kde το φωνάζουμε kubuntu. Είναι πλήρες(!), τεχνολογικά προηγμένο(!), και εύκολο στην χρήση από οποιονδήποτε(!). Στα αποθετήρια του ubuntu υπάρχουν διαθέσιμες κυριολεκτικά χιλιάδες εφαρμογές σχεδόν για οτιδήποτε(!) … για επαγγελματική, επιστημονική, εκπαιδευτική, και οικιακή χρήση. Τόσο το ubuntu όσο και οι εφαρμογές του είναι Ελεύθερο Λογισμικό / Λογισμικό Ανοιχτού Κώδικα (ΕΛ/ΛΑΚ), δηλαδή διατίθενται ελεύθερα, και στην Ελλάδα υποστηρίζονται από την άτυπη αλλά πολύ δραστήρια κοινότητα ubuntu-gr. Περισσότερα στο `http://www.ubuntu-gr.org`.

Η κοινότητα ubuntu-gr

Η κοινότητα ubuntu-gr ανήκει στα μέλη της και είναι ανοιχτή σε όλους! Είναι το μέρος όπου έμπειροι και άπειροι(!) χρήστες συζητάνε ό,τι τους απασχολεί, ιδέες, ερωτήματα, πρακτικά ζητήματα, οργανωτικά θέματα, και κυρίως τεχνικά προβλήματα. Αποτελείται από ανθρώπους με εμπειρία στην πληροφορική αλλά κυρίως από απλούς χρήστες, οι οποίοι εθελοντικά συμμετέχουν i) στην δημιουργία-ανάπτυξη του λογισμικού, ii) στην μετάφρασή του στην ελληνική γλώσσα, iii) στην προώθηση-διάδοση του στην Ελλάδα, και κυρίως iv) στην παροχή αμεσότατης(!) και υψηλής ποιότητας(!) τεχνικής υποστήριξης σε άλλους ελληνόφωνους χρήστες. Λειτουργεί με αυτό-οργάνωση και προσπαθούμε οι αποφάσεις να λαμβάνονται όσο το δυνατόν πιο δημοκρατικά από εκείνους που προσφέρουν-δραστηριοποιούνται συστηματικά. Η ελληνική κοινότητα του Ubuntu διαθέτει μέχρι στιγμής φόρουμ, λίστα ηλ. ταχυδρομείου, κανάλι συζητήσεων τύπου IRC, καθώς και το περιοδικό Ubuntistas. Για όλα αυτά υπάρχουν οδηγίες και links στο `http://www.ubuntu-gr.org`.

Το περιοδικό ubuntistas

Το Ubuntistas, το ηλεκτρονικό περιοδικό της ελληνικής κοινότητας του ubuntu (ubuntu-gr), κυκλοφορεί ελεύθερα κάθε δίμηνο, με πρώτο τεύχος του Νοεμβρίου - Δεκεμβρίου 2008. Περιέχει νέα, πληροφορίες, συνεντεύξεις, παρουσιάσεις, οδηγούς, και άρθρα σχετικά με το ubuntu. Το περιοδικό είναι ανοιχτό σε όλους, όπως και το GNU/Linux! Ο καθένας μπορεί να συμμετέχει ενεργά στην δημιουργία του, να αρθρογραφήσει, να προτείνει ιδέες και να κάνει τις επισημάνσεις / παρατηρήσεις του.

του Χατζηπαντελή Παντελή

Νέα & Ειδήσεις

... από τον κόσμο του linux και όχι μόνο!

LibreOffice, η μετεξέλιξη του OpenOffice.org

Στις 28 Σεπτεμβρίου 2010 τα τρέχοντα μέλη του OpenOffice.org Community ανακοίνωσαν τη δημιουργία του The Document Foundation και τη διάθεση ενός νέου fork του OpenOffice.org με την ονομασία LibreOffice. Η βασική αιτία για την ίδρυση του νέου οργανισμού Document Foundation ήταν η αβεβαιότητα που δημιουργήθηκε στους χρήστες OO.org μετά την εξαγορά της Sun από την Oracle. Αν και η Oracle δεν έχει δείξει σημάδια "εγκατάλειψης" της δημοφιλούς σουίτας γραφείου, έγινε φανερό σε όλους το γεγονός ότι μια μελλοντική εξαγορά ή αλλαγή πολιτικής θα μπορούσε να αποβεί μοιραία για την τύχη της πιο δημοφιλούς σουίτας γραφείου ανοικτού λογισμικού.

Ο κύριος σκοπός του Document Foundation είναι η δημιουργία μιας σουίτας γραφείου, ανεξάρτητης, απρόσβλητης από τις επιταγές μίας και μόνο εταιρίας (τουλάχιστον θεωρητικά) και ελεύθερης από πιθανούς περιορισμούς πνευματικών δικαιωμάτων. Το Document Foundation υποστηρίζεται ήδη από το FSF, την Google, τη Novell, τη RedHat, την Canonical, το OSI, το Gnome Foundation, και μία πλειάδα από οργανισμούς που στο παρελθόν υποστήριζαν το OO.org.

Ήδη έχει ανακοινωθεί ότι το LibreOffice θα ενσωματωθεί στις μελλοντικές εκδόσεις Ubuntu, με τις διανομές που πρόσκεινται στην RedHat να αναμένονται να πράξουν το ίδιο σύντομα. Ο μεγάλος απών από το "πάρτι" είναι για την ώρα η Oracle. Η εταιρία φάνηκε μάλλον απροετοίμαστη για αυτή την εξέλιξη και μέχρι τα τέλη Οκτωβρίου δεν είχε επίσημα ανακοινώσει τις προθέσεις της. Τελικά, η Oracle, αξιώνοντας ότι υπήρχαν αντικρουόμενα συμφέροντα μεταξύ των δύο έργων, πρότεινε/απαίτησε την παραίτηση όσων developers του OO.org είχαν αναμειχθεί στην ανάπτυξη του LibreOffice, οδηγώντας σε άμεση παραίτηση 33 άτομα.

Τεχνικά, το LibreOffice είναι η συνέχιση του OpenOffice.org. Η παρούσα έκδοση άλλωστε είναι ουσιαστικά ένα re-branding του OpenOffice.org 3.3.0. Στον άμεσο ορίζοντα οι νέες εκδόσεις αναμένονται να ενσωματώσουν πολλά από τα "καλούδια" που προσφέρουν άλλα forks του OO.org, όπως το Go-oo και το NeoOffice. Για την ώρα το LibreOffice είναι διαθέσιμο μονάχα στην Αγγλική γλώσσα. Η μεταφραστική διαδικασία πάντως έχει ήδη αρχίσει και οι πρώτες "εντόπιες" εκδόσεις αναμένονται σύντομα, καθώς ουσιαστικά αποτελούν περισσότερο ελέγχους συμβατότητας παρά αυτούσια μεταφραστικά έργα.

Το Document Foundation και το LibreOffice προέκυψαν από την πεποίθηση ότι η κουλτούρα και το έργο που δημιουργείται από ένα ανεξάρτητο και αξιοκρατικό ίδρυμα μπορεί να δημιουργήσει το βέλτιστο περιβάλλον και για την ανάπτυξη λογισμικού αλλά και για την τελική εμπειρία ενός χρήστη. Το LibreOffice ξεκινάει λοιπόν τη σταδιοδρομία του με τις καλύτερες προθέσεις. Ευχή όλων μας είναι αυτές οι προθέσεις να επαληθευτούν.

του Κωστάρα Γιάννη

Εισαγωγή στο Mercurial

Ένα σύστημα διαχείρισης εκδόσεων.

Από τις αρχές της ιστορίας του προγραμματισμού κρίθηκε αναγκαία η ύπαρξη εκδόσεων των πηγαίων αρχείων. Ο έλεγχος των διαφόρων εκδόσεων γινόταν αρχικά από τον ίδιο τον προγραμματιστή. Γρήγορα έγινε φανερό ότι ακόμα και ο έλεγχος των εκδόσεων ενός μόνο αρχείου υπόκειται στην πιθανότητα λάθους, και γι' αυτό το λόγο αναπτύχθηκαν εργαλεία για την αυτοματοποίηση της διαδικασίας.

Τα πρώτα συστήματα διαχείρισης εκδόσεων μπορούσαν να διαχειριστούν έναν μόνο χρήστη. Το πρώτο τέτοιο σύστημα ήταν το SCCS (Source Code Control System) στις αρχές της δεκαετίας του 1970 και το RCS (Revision Control System) το 1982. Σιγά σιγά όμως εξελίχθηκαν ώστε να μπορούν να διαχειρίζονται πολλά αρχεία και πολλούς χρήστες ταυτόχρονα. Τα περισσότερα συστήματα διαχείρισης εκδόσεων (revision control systems) ακολουθούν το μοντέλο διακομιστή-πελάτη, όπου υπάρχει ένας κεντρικός διακομιστής που φιλοξενεί τις διάφορες εκδόσεις. Τα πιο γνωστά συστήματα διαχείρισης εκδόσεων ακολουθούν αυτό το μοντέλο, όπως το CVS (Concurrent Version Control) και ο διάδοχός του Subversion, το ClearCase, το Perforce, το Merant PVCS κ.ά.

Την τελευταία δεκαετία έχουν αναπτυχθεί και τα λεγόμενα κατανεμημένα συστήματα διαχείρισης εκδόσεων, όπως το Git, παιδί του Linux που δημιουργήθηκε για την ανάπτυξη του πυρήνα του Linux από τους ανά τον κόσμο προγραμματιστές, και το Mercurial. Αυτά τα συστήματα δεν περιορίζονται σ' έναν διακομιστή, ο οποίος μπορεί να εξυπηρετεί μόνο μέχρι έναν ορισμένο αριθμό από χρήστες, ενώ όταν αυτός 'κρασάρει', τότε σταματούν και οι ενημερώσεις των εκδόσεων. Αντίθετα, στα κατανεμημένα συστήματα εκδόσεων, κάθε χρήστης έχει το δικό του τοπικό αποθετήριο (repository) των εκδόσεων, κι όταν χρειαστεί να συνεργαστεί με άλλους χρήστες, τότε γίνεται συγχώνευση των αποθετηρίων τους. Με άλλα λόγια, κάθε χρήστης έχει το δικό του backup και δεν απαιτείται συνεχής σύνδεση στο δίκτυο για να γίνει commit.

Οι λόγοι χρήσης συστημάτων διαχείρισης εκδόσεων είναι πολλοί:

- Διαχειρίζονται το ιστορικό και την εξέλιξη του έργου μας. Καταγράφουν ποιος έκανε την αλλαγή, τι αλλαγή έκανε, πότε την έκανε και γιατί την έκανε.
- Βοηθάνε στη συνεργασία με άλλους (απομακρυσμένους) προγραμματιστές.
- Επιτρέπουν την παράλληλη ανάπτυξη διαφορετικών εκδόσεων του ίδιου αρχείου.
- Μας βοηθάνε να επανέλθουμε από μια λάθος εξέλιξη του έργου μας πίσω στην τελευταία έκδοση που δούλευε.

Παράλληλα, το Mercurial προσφέρει πολλά πλεονεκτήματα συγκριτικά με άλλα συστήματα διαχείρισης εκδόσεων:

- Είναι εύκολο στην εκμάθηση.
- Είναι 'ελαφρύ', δηλαδή δεν απαιτεί πολλούς πόρους.
- Είναι πιο γρήγορο και αξιόπιστο απ' τους ανταγωνιστές του.
- Τροποποιείται εύκολα, ανάλογα με τις ανάγκες μας.
- Δεν μειώνεται η απόδοσή του όσο κι αν αυξηθούν οι χρήστες ή ο όγκος των αρχείων.

- Είναι ανοικτού κώδικα και φυσικά δωρεάν.

Επίσης:

- Το Mercurial υπερτερεί σε σχέση με το Subversion σε θέματα απόδοσης, αλλά και στο θέμα της συγχώνευσης, στο οποίο το Subversion υστερεί.
- Είναι πολύ πιο απλό απ' το πολύπλοκο Git και δεν χρειάζεται συντήρηση.
- Φυσικά υπερτερεί και σε σχέση με το προβληματικό CVS (όσοι το έχετε χρησιμοποιήσει καταλαβαίνετε τον χαρακτηρισμό 'προβληματικό'), τόσο σε επιδόσεις όσο και σε αξιοπιστία.

Αν λοιπόν χρειάζεστε ένα αξιόπιστο σύστημα διαχείρισης εκδόσεων, ή δεν είστε ευχαριστημένοι απ' αυτό που χρησιμοποιείτε, τότε θα πρέπει να δοκιμάσετε το Mercurial. Γι' αυτό, διαβάστε παρακάτω.

Εγκατάσταση

Η εγκατάσταση του Mercurial στο Ubuntu ακολουθεί την πεπατημένη:

```
$ sudo apt-get install mercurial
```

Βασικές εντολές

Η εντολή είναι hg, όπως ονομάζεται το χημικό στοιχείο “υδράργυρος” στον περιοδικό πίνακα. Ας δούμε τι μπορούμε να κάνουμε με το Mercurial:

```
$ hg version
Mercurial Distributed SCM (version 1.4.3)


   Copyright (C) 2005-2010 Matt Mackall <mpm@selenic.com> and others
This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Οι παρακάτω εντολές παρέχουν λεπτομερή βοήθεια για τις εντολές του Mercurial:

```
$ hg help
$ hg help <command>
$ hg -v help
$ man hg
$ man hgignore
$ man hgrc
```

Επίσης, μπορείτε να βρείτε διάφορα παραδείγματα για εργαλεία κλπ. στο φάκελο: /usr/share/doc/mercurial-common/examples/. Το Mercurial δουλεύει με αποθετήρια (repositories). Αν γνωρίζετε ήδη κάποιο αποθετήριο Mercurial στο οποίο θέλετε να δουλέψετε (π.χ. κάποιο έργο ανοικτού κώδικα), τότε μπορείτε να δημιουργήσετε ένα αντίγραφό του στο δίσκο σας με την εντολή:

```
$ hg clone <URL of Mercurial repository>
```

η οποία, όπως λέει και τ' όνομά της, δημιουργεί έναν τοπικό κλώνο (αντίγραφο) του απομακρυσμένου αποθετηρίου. Π.χ. ας αντιγράψουμε το repository `http://hg.serpentine.com/tutorial/hello`

```
$ hg clone http://hg.serpentine.com/tutorial/hello
destination directory: hello
requesting all changes
adding changesets
adding manifests
adding file changes
added 5 changesets with 5 changes to 2 files
updating working directory
2 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

Όπως βλέπετε, η εντολή δημιούργησε τον φάκελο hello μέσα στον φάκελο από όπου την καλέσατε. Αν π.χ. καλέσατε την εντολή ενώ βρισκόσασταν στο φάκελο Documents, δημιουργήθηκε ο φάκελος Documents/hello μέσα στον οποίο εμφανίστηκαν δυο αρχεία.

```
$ ls hello
Makefile hello.c
```

Παρατηρήστε όμως ότι έχει δημιουργηθεί κι ένας κρυφός φάκελος, ο .hg, ο οποίος είναι το αποθετήριό σας αυτό καθ' αυτό. Εκεί μέσα αποθηκεύονται όλες οι αλλαγές που συντελούνται στα αρχεία σας.

```
$ hg history          // ή hg log
changeset:   4:2278160e78d4
tag:         tip
user:        Bryan O'Sullivan <bos@serpentine.com>
date:        Sat Aug 16 22:16:53 2008 +0200
summary:     Trim comments.

changeset:   3:0272e0d5a517
user:        Bryan O'Sullivan <bos@serpentine.com>
date:        Sat Aug 16 22:08:02 2008 +0200
summary:     Get make to generate the final binary from a .o file.

changeset:   2:fef857204a0c
user:        Bryan O'Sullivan <bos@serpentine.com>
date:        Sat Aug 16 22:05:04 2008 +0200
summary:     Introduce a typo into hello.c.

changeset:   1:82e55d328c8c
user:        mpm@selenic.com
date:        Fri Aug 26 01:21:28 2005 -0700
summary:     Create a makefile

changeset:   0:0a04b987be5a
user:        mpm@selenic.com
date:        Fri Aug 26 01:20:50 2005 -0700
summary:     Create a standard "hello, world" program
```

Να έχετε υπόψιν σας ότι ο ακέραιος πριν το : στο changeset ή σύνολο αλλαγών (λέγεται αριθμός έκδοσης - revision) δεν είναι μοναδικός, καθώς κάποιος άλλος προγραμματιστής μπορεί κι αυτός να 'χει δημιουργήσει την δικιά του έκδοση 4 που να διαφέρει απ' τη δική σας. Ο αριθμός όμως μετά το : είναι μοναδικός (λέγεται αναγνωριστικό changeset) σε όλα τα αποθετήρια του έργου. Έτσι, στο παραπάνω παράδειγμα το 4 είναι ο αριθμός έκδοσης, ενώ το 2278160e78d4 είναι το αναγνωριστικό του changeset κι είναι μοναδικό σε όλα τα αποθετήρια.

```
$ hg log -r 2:3
changeset:   3:0272e0d5a517
user:        Bryan O'Sullivan <bos@serpentine.com>
date:        Sat Aug 16 22:08:02 2008 +0200
summary:     Get make to generate the final binary from a .o file.

changeset:   2:fef857204a0c
user:        Bryan O'Sullivan <bos@serpentine.com>
date:        Sat Aug 16 22:05:04 2008 +0200
summary:     Introduce a typo into hello.c.
```

Παρατηρήστε ότι, όταν δώσατε hg log πιο πάνω, το πιο πρόσφατο changeset εμφανίστηκε με μια ετικέτα (tag) με όνομα tip.

```
$ hg tip //   hg head
changeset:  4:2278160e78d4
tag:  tip
user:  Bryan O'Sullivan <bos@serpentine.com>
date:  Sat Aug 16 22:16:53 2008 +0200
summary:  Trim comments.
```

Το head είναι ένας δείκτης στην πιο πρόσφατη υποβολή αλλαγών που κάνατε. Στο Mercurial μπορείτε όμως να δημιουργήσετε κλάδους (branches), και καθένας από αυτούς θα 'χει το δικό του head. Το head με τον μεγαλύτερο αριθμό έκδοσης λέγεται tip. Στο παράδειγμά μας δεν έχουμε δημιουργήσει ακόμα κλάδους, οπότε head = tip. Και φυσικά με την εντολή:

```
$ hg tag release1
```

μπορείτε να δώσετε μια ετικέτα στην τελευταία αλλαγή που κάνατε. Μπορείτε φυσικά να κρατήσετε ανέπαφο το αρχικό repository και να δουλέψετε μ' ένα αντίγραφό του.

```
$ hg clone hello hello-orig
```

Αν κάνατε κάποια αλλαγή, τότε:

```
$ hg status
M hello.c
$ hg diff
diff -r 2278160e78d4 hello.c
```

Υποβολή των αλλαγών

Προτού όμως μπορέσουμε να υποβάλουμε τις αλλαγές μας πίσω στο αποθετήριο, σ' ένα νέο changeset, θα πρέπει το Mercurial να γνωρίζει το email μας και ένα όνομα χρήστη. Το πιο εύκολο είναι να δημιουργήσουμε το αρχείο ~/.hgrc με τα εξής περιεχόμενα:

```
[ui]
username = hawk <nobody@nowhere.gr>
[email]
from = hawk <nobody@nowhere.gr>
```

όπου φυσικά στη θέση του hawk δίνετε το δικό σας όνομα χρήστη και email. Στη συνέχεια, υποβάλλετε τις αλλαγές σας με την εντολή:

```
$ hg commit
This is where I type my commit comment.
HG: Enter commit message.  Lines
beginning with 'HG:' are removed.
HG: --
HG: user: hawk <nobody@nowhere.gr>
HG: branch 'default'
HG: changed hello.c
```

Ενημέρωση αποθετηρίου

Προτού ενημερώσετε το αποθετήριό σας με τις αλλαγές που έχουν γίνει στο μεταξύ στο αρχικό αποθετήριο, από το οποίο δημιουργήσατε το αντίγραφό σας, (ή από κάποιο άλλο αποθετήριο), χρησιμοποιήστε την εντολή:

```
$ hg outgoing <repository>
```

για να δείτε τι αλλαγές έχουν συμβεί, όπου <repository> μπορεί να είναι π.χ. `http://hg.serpentine.com/tutorial/hello`. Αν δεν βλέπετε κάποιο πρόβλημα, τότε:

```
$ hg pull <repository>
```

για να ενημερώσετε τον αποθετήριό σας με τις αλλαγές που έχουν γίνει στο απομακρυσμένο αποθετήριο. Το Mercurial όμως ξεχωρίζει την ενημέρωση του αποθετηρίου από την ενημέρωση του καταλόγου με τ' αρχεία σας αυτού καθ' αυτού. Αν θέλετε να ενημερώσετε τα αρχεία σας, τότε θα πρέπει να δώσετε μια ακόμα εντολή:

```
$ hg update
```

Για να δείτε αν όντως ενημερώθηκε ο κατάλογος με τα αρχεία σας:

```
$ hg parents
changeset: 5:12efb75cbece
tag: tip
user: hawk <nobody@nowhere.gr>
date: Tue June 29 06:44:49 2010 +0300
summary: Added an extra line of output
```

Ο λόγος που υπάρχει και η pull και η update είναι ότι αν για κάποιο λόγο δεν είστε ευχαριστημένοι με τις αλλαγές, μπορείτε πάντα να γυρίσετε σε μια προηγούμενη έκδοση, π.χ.:

```
$ hg update 3
```

Πάντως, μπορείτε να συγχωνεύσετε τα δυο αυτά βήματα με την εντολή:

```
$ hg pull -u
```

Αντίστοιχα με τις incoming και pull, υπάρχουν και οι outgoing και push, αν θέλουμε να στείλουμε τις αλλαγές μας σε κάποιο άλλο αποθετήριο.

```
$ hg incoming <repository>
$ hg push <repository>
```

Άλλες χρήσιμες εντολές:

```
$ hg addremove  // προσθέτει όλα τα νέα
αρχεία και διαγράφει όλα όσα λείπουν
$ hg copy  // σημειώνει τα αρχεία ως
αντιγραμμένα, για το επόμενο commit
$ hg rename  // σημειώνει τα αρχεία ως
μετονομασθέντα, για το επόμενο commit
$ hg forget, hg remove // ξεχνά ή διαγράφει
τα συγκεκριμένα αρχεία
$ hg revert   // επαναφέρει αρχεία/
καταλόγους σε μια προηγούμενη κατάσταση, αν
δεν είχε προηγηθεί commit
$ hg backout    // επαναφέρει σε προηγούμενο
changeset, αν είχε γίνει commit
```

Δημιουργία αποθετηρίου

Αν θέλετε να δημιουργήσετε ένα νέο αποθετήριο στον τρέχοντα κατάλογο:

```
$ hg init myproject
```

Στη συνέχεια, μπορείτε να αντιγράψετε ή να δημιουργήσετε κάποια αρχεία στο νέο αποθετήριο, και να ενημερώσετε το Mercurial για τις ενέργειές σας, δίνοντας:

```
$ cd myproject
$ hg add
$ hg status
A goodbye.c
A hello.c
$ hg commit -m 'Initial commit'
```

Συγχώνευση

Μέχρις εδώ όλα καλά. Τι γίνεται όμως άμα τόσο εσείς όσο κι ένας συνάδελφος αλλάξατε το ίδιο αρχείο; Ας υποθέσουμε ότι εκτελούμε την εντολή:

```
$ hg pull <another-repository>
pulling from <another-repository>
searching for changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 1 changes to
1 files (+1 heads)
(run 'hg heads' to see heads, 'hg
merge' to merge)
```

Η τελευταία αλλαγή που κάναμε λέγεται κεφαλή (head). Αν όμως κάποιος άλλος έκανε κι αυτός μια αλλαγή στο δικό του αποθετήριο, την οποία τραβήξαμε, όπως με την παραπάνω εντολή, στο δικό μας, είναι κι αυτή μια άλλη κεφαλή.

```
$ hg heads
changeset: 6:12efb75cbece
tag: tip
parent: 4:2278160e78d4
user: hawk <nobody@nowhere.gr>
date: Tue June 29 06:44:49 2010 +0300
summary: Added an extra line of output

changeset: 5:cbfc9ee6ea99
user: hawk <nobody@nowhere.gr>
date: Tue June 29 06:44:52 2010 +0300
summary: A new hello for a new day.
$ hg update
abort: crosses branches (use 'hg
merge' or 'hg update -C')
$ hg merge
merging hello.c
0 files updated, 1 files merged, 0
files removed, 0 files unresolved
(branch merge, don't forget to commit)
$ hg parents
```

```
changeset:  5:cbfc9ee6ea99
user:  hawk <nobody@nowhere.gr>
date:  Tue June 29 06:44:52 2010 +0300
summary:  Bug fix.

changeset:  6:12efb75cbece
tag:  tip
parent:  4:2278160e78d4
user:  hawk <nobody@nowhere.gr>
date:  Tue June 29 06:44:49 2010 +0300
summary:  Added an extra line of output
$ hg commit -m 'Merged changes'
$ hg tip
changeset:  7:41ec93b29030
tag:  tip
parent:  5:cbfc9ee6ea99
parent:  6:12efb75cbece
user:  hawk <nobody@nowhere.gr>
date:  Tue June 29 06:44:52 2010 +0300
summary:  Merged changes
```

Στο παραπάνω παράδειγμα δεν υπήρξαν συγκρούσεις (conflicts), δηλ. δεν αλλάχθηκαν ίδιες γραμμές κώδικα. Τι γίνεται όμως στην περίπτωση που δυο προγραμματιστές αλλάξουν τις ίδιες γραμμές κώδικα; Στην περίπτωση αυτή, το Mercurial σηκώνει τα χέρια ψηλά, τρέχει ένα κατάλληλο πρόγραμμα συγχώνευσης (το οποίο ορίζετε στη μεταβλητή συστήματος HGMERGE), και σας ζητάει να κάνετε εσείς τη συγχώνευση. Υπάρχουν πολλά εργαλεία συγχώνευσης στο Ubuntu, όπως τα merge, diff, kdiff3, meld, gvimdiff, vimdiff. Μόλις κάνετε τις συγχωνεύσεις, δίνετε:

```
$ hg resolve -m hello.c
$ hg commit -m 'Conflicts merged'
$ hg tip
changeset:  8:cef275730d6e
tag:  tip
parent:  7:4a31891298cb
parent:  6:44d8436f9b83
user:  hawk <nobody@nowhere.gr>
date:  Tue June 29 06:44:53 2010 +0300
summary:  Conflicts merged
```

Το Mercurial συγχωνεύει τις τρεις παραπάνω εντολές:

```
hg pull -u
hg merge
hg commit -m 'Merged remote changes'
```

σε μία

```
hg fetch
```

φτάνει να προσθέσετε στο αρχείο /.hgrc τις παρακάτω γραμμές:

```
[extensions]
fetch =
```

Εύρεση λαθών

Αργά ή γρήγορα, λάθη θα βρουν το δρόμο τους για να χαλάσουν τον κώδικά σας. Το Mercurial έχει 2 τρόπους για να ξετρυπώνει τα λάθη:

```
$ hg annotate hello.c
```

Εμφανίζει το 'βλαβερό' αρχείο, προσθέτοντας τον αριθμό έκδοσης μπροστά από κάθε γραμμή. Υπάρχει όμως και η bisect.

```
$ hg bisect -b tip // σημείωσε το tip ως 'κακό'
$ hg bisect -g release1 // έστω ότι η release1 δούλευε σωστά
Testing changeset 7:912c42b59126 (3 changesets remaining, 1 tests)
1 files updated, 0 files merged, 0 files removed, 0 files unresolved
$ make // δοκιμάστε αν χτίζεται
```

Σημειώστε το ως 'καλό' με την hg bisect -g, ή ως 'κακό' με την hg bisect -b. Επαναλάβετε τη διαδικασία μέχρις ότου βρείτε ποιο commit έκανε τη ζημιά. Μπορείτε επίσης να αυτοματοποιήσετε τη διαδικασία δίνοντας:

```
$ hg bisect -c make
Changeset 7:912c42b59126: bad
The first bad revision is:
changeset:  7:912c42b59126
user:  hawk <nobody@nowhere.gr>
date:  Sat Sep 18 19:50:50 2010 +0200
summary:  Added tag release1 for changeset 4c69a3c69d2c
```

TortoiseHG

Αν σας ενοχλεί η γραμμή εντολών, μην ανησυχείτε. Από το Ubuntu Software Center μπορείτε να εγκαταστήσετε τα tortoisehg και tortoisehg-nautilus. Όταν τα εγκαταστήσετε, επανεκινήστε το Ubuntu. Το TortoiseHG ενσωματώνεται στο μενού που εμφανίζεται με δεξί κλικ στον Nautilus και εμφανίζει πολλές από τις παραπάνω εντολές με γραφικό τρόπο. Παράλληλα, χρησιμοποιείται και από τη γραμμή εντολών:

```
$ hgtk
```

(συνεχίζεται στη σελίδα 14)

του Κωστάρα Γιάννη

Εισαγωγή στην Κρυπτογραφία με τη γλώσσα Java – Μέρος 2ο

Κρυπτογραφία Δημόσιου Κλειδιού.

Στο προηγούμενο τεύχος του Ubuntistas μιλήσαμε για τη Συμμετρική Κρυπτογραφία και είδαμε ένα πρόγραμμα σε Java χρησιμοποιώντας τις βιβλιοθήκες Java Cryptography Extension (JCE). Στο 2ο μέρος θα μιλήσουμε για την Ασύμμετρη Κρυπτογραφία ή Κρυπτογραφία Δημόσιου Κλειδιού.

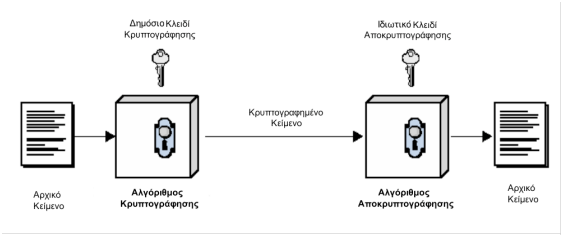
Κρυπτογραφία δημόσιου κλειδιού

Το πρόβλημα που έχουμε με την κρυπτογραφία δημόσιου κλειδιού είναι πώς να διανείμουμε το μυστικό κλειδί χωρίς να υποπέσει σε άσχετα χέρια. Θα μπορούσαμε φυσικά να δώσουμε το μυστικό κλειδί στον παραλήπτη χέρι με χέρι, μέσω ασφαλούς τηλεφωνικής επικοινωνίας, μέσω ασφαλούς αλληλογραφίας κλπ. Αλλά, τέτοιες μέθοδοι δεν είναι ούτε πρακτικές ούτε πολλές φορές εφικτές. Οι πιο γνωστοί αλγόριθμοι δημόσιου κλειδιού είναι οι [1,3,4]:

- Diffie-Hellman
- RSA
- El Gamal

Το 1976, μια επαναστατική μελέτη άνοιξε νέους ορίζοντες στην κρυπτογραφία, ορίζοντας την ασύμμετρη κρυπτογραφία ή κρυπτογραφία δημόσιου κλειδιού. Ο αλγόριθμος Diffie-Hellman παίρνει τ' όνομά του από τους Whitfield Diffie και Martin Hellman, και βασίζεται στη δυσκολία επίλυσης του διακριτού λογαριθμικού προβλήματος. Ο αλγόριθμος RSA δημοσιεύθηκε το 1977 από τους Ronald Rivest, Adi Shamir, και Leonard Adleman. Η ασφάλεια του αλγορίθμου έγκειται στη δυσκολία παραγοντοποίησης μεγάλων πρώτων αριθμών. (Ένας αριθμός είναι πρώτος αν διαιρείται μόνο από το 1 και τον εαυτό του). Ο αλγόριθμος El Gamal (1985), τέλος, είναι μια παραλλαγή του Diffie Hellman.

Η λειτουργία των αλγορίθμων δημοσίου κλειδιού φαίνεται στο ακόλουθο σχήμα.



Σχήμα 1: Λειτουργία του αλγορίθμου δημοσίου κλειδιού

Ο αποστολέας χρησιμοποιεί το δημόσιο κλειδί του παραλήπτη για να κρυπτογραφήσει το μήνυμα και το στέλνει στον παραλήπτη. Ο παραλήπτης χρησιμοποιεί το ιδιωτικό του κλειδί για να αποκρυπτογραφήσει το μήνυμα. Μόνο ο παραλήπτης (ο κάτοχος του ιδιωτικού κλειδιού) μπορεί να αποκρυπτογραφήσει το κρυπτομήνυμα. Γνωρίζοντας κάποιος το δημόσιο κλειδί δεν μπορεί να συνάγει καμία πληροφορία όσον αφορά το ιδιωτικό κλειδί.

Το πιο γνωστό παράδειγμα κρυπτογραφίας δημόσιου κλειδιού είναι το PGP (Pretty Good Privacy). Οι χρήστες PGP παράγουν ένα ζεύγος κλειδιών και μεταφορτώνουν το δημόσιο κλειδί σ' ένα διακομιστή για να μπορεί να το χρησιμοποιήσει ο καθένας. Όποιος θέλει να στείλει ένα κρυπτογραφημένο μήνυμα στον χρήστη του δημόσιου κλειδιού, το χρησιμοποιεί για να κρυπτογραφήσει το μήνυμά του με αυτό και στη συνέχεια στέλνει το κρυπτογραφημένο μήνυμα στον παραλήπτη μέσω ηλεκτρονικού ταχυδρομείου. Ο παραλήπτης είναι ο μόνος που μπορεί να αποκρυπτογραφήσει το κρυπτομήνυμα με το ιδιωτικό του κλειδί. Φυσικά το ερώτημα παραμένει. Πώς μπορείς να είσαι σίγουρος ότι το δημόσιο κλειδί που θα βρεις στον διακομιστή είναι

όντως του ισχυριζόμενου παραλήπτη;

Το πρόγραμμα που μπορείτε να κατεβάσετε από `http://files.ubuntu-gr.org/ubuntistas/files/java_cryptography.zip`, επιδεικνύει ένα παράδειγμα κρυπτογραφίας δημόσιου κλειδιού. Το πρόγραμμα μοιάζει πολύ με αυτό της συμμετρικής κρυπτογραφίας του προηγούμενου τεύχους, το οποίο μπορείτε επίσης να κατεβάσετε από `http://files.ubuntu-gr.org/ubuntistas/files/java_cryptography.zip`.

Για να μεταγλωτίσετε το πρόγραμμα απαιτείται να προσθέσετε τη βιβλιοθήκη commons-codec.jar στο classpath, αν δεν το έχετε κάνει ήδη, την οποία μπορείτε να κατεβάσετε από την ιστοσελίδα `http://commons.apache.org/codec/`. Η βιβλιοθήκη αυτή περιέχει την κλάση Base64, η οποία μας επιτρέπει να αναπαριστούμε τους κώδικες σε μια πιο φιλική μορφή (την Base64). Παρόλ' αυτά, το πρόγραμμα δουλεύει μια χαρά και χωρίς αυτήν την κλάση, την οποία μπορείτε να βγάλετε εύκολα. Μεταγλωττίστε και εκτελέστε το πρόγραμμα (απαιτείται JDK 1.6 ή νεώτερο). Ένα παράδειγμα εκτέλεσης φαίνεται παρακάτω:

```
$ java -cp .:lib/commons-codec-1.4.jar
gr.ubuntistas.issue10.SimpleASymme
tricExample
This is a long message!
Plain text input: This is a long
message!
Cipher text: a7NuxHv+FVrHTgRTNGP8dwf
wR47h7Y7Hfllzx80rp+/mR/aWDPZr0DZ/AFV
I0AjRVJqvsEd1taTo mjXIjUv62RE5YAhlGk
m9WeZlJUFL56UJ5uQErnkvMBQs0cRf4hFxZj
ExsG7Htilnc84oqhYOJNIyIpQVWmRPhSnZC
DbXBNU=

Plain text output: This is a long
message!
```

Παρατηρήστε ότι το μέγεθος του κρυπτοκειμένου είναι κατά πολύ μεγαλύτερο απ' αυτό της συμμετρικής κρυπτογραφίας. Το πρόγραμμα είναι σχεδόν πανομοιότυπο με το πρόγραμμα συμμετρικής κρυπτογραφίας που παρουσιάσαμε στο προηγούμενο τεύχος, με τις ακόλουθες αλλαγές:

- αντί για SecretKey, η SimpleASymmetricExample χρησιμοποιεί KeyPair, μια κλάση που αποθηκεύει το δημόσιο και το ιδιωτικό κλειδί. Ως επί των πλείστων, η generateKey() έχει αλλάξει στην generateKeyPair()
- στην ecrypt() η cipher.init(Cipher.ENCRYPT_MODE, keypair.getPublic()) χρησιμοποιεί το δημόσιο κλειδί για να κρυπτογραφήσει το μήνυμα, ενώ
- στην decrypt() η cipher.init(Cipher.DECRYPT_MODE, keypair.getPrivate()) χρησιμοποιεί το ιδιωτικό κλειδί για να μπορέσει να αποκρυπτογραφήσει το μήνυμα.

Κατά τ' άλλα, το πρόγραμμα είναι όμοιο μ' αυτό του προηγούμενου τεύχους!

Δυστυχώς, ο πάροχος SunJCE δεν έχει υλοποιήσει άλλους αλγορίθμους πέραν των RSA και Diffie-Hellman (DH). Αν θέλετε να χρησιμοποιήσετε padding, όπως π.χ. RSA/None/PKCS1Padding ή RSA/None/OAEPWithSHA1AndMGF1Padding, τότε θα πρέπει να χρησιμοποιήσετε άλλον πάροχο, όπως π.χ. τον BouncyCastle.

Θα χρειαστεί να κάνετε τις ακόλουθες αλλαγές. Καταρχάς, κατεβάστε και προσθέστε στο φάκελο lib τη βιβλιοθήκη bcprov-jdk16-xxx.jar από την ιστοσελίδα της BouncyCastle. Στην αρχή της main(), προσθέστε τη γραμμή:

```
Security.addProvider(new BouncyCastle
Provider());
```

Θα πρέπει να προστεθεί ο πάροχος «BC» στις παρακάτω γραμμές:

```
cipher = Cipher.getInstance(algorithm,
"BC");
keypairgenerator = KeyPairGenerator.
getInstance(algorithm,"BC");
```

Επίσης, η generateKeyPair() δεν καταλαβαίνει π.χ. το RSA/None/PKCS1Padding, οπότε της περνάτε μόνο το RSA με την παρακάτω γραμμή κώδικα:

```
keypair = generateKeyPair(algorithm.
indexOf('/') == -1 ? algorithm :
algorithm.substring(0, algorithm.
indexOf('/')));
```

Μπορείτε να κατεβάσετε την κλάση SimpleASymmetricExampleBC από `http://files.ubuntu-gr.org/ubuntistas/`

files/java_cryptography.zip. Μεταγλωττίστε και εκτελέστε το πρόγραμμα. Ένα παράδειγμα εκτέλεσης φαίνεται παρακάτω:

```
$ java -cp .:lib/commons-codec-1.4.jar:
lib/bcprov-jdk16-145.jar gr.ubuntistas.
issue10.SimpleASymmetricExampleBC
This is a long message!
Plain text input: This is a long
message!
Cipher text: Qk0xXw+DFxOeIccZQYza5xH3
kp9AvKA9GYxIGP21JzTuW0saagv+0oKRoJy60
ScHzC4faRAGGvQr y/VVACGZnqVucxkJgAuW+
SFbjumd3+EwCk84ivl5Y3ZPVB/TdHat5noiN6
5EO/iIggdc7gXIc+rZ 1Q8oWMFFYPCrQSZCtg
g=

Plain text output: This is a long
message!
```

Κι ένα παράδειγμα χρήσης του αλγορίθμου El Gamal (αλλάζοντας μόνο το «RSA» σε «El Gamal» στη μέθοδο main()!):

```
Plain text input: This is a long
message!
Cipher text: XMXaHtlbDJ72N4nocHLNbkFh
PB7fKheMt3pjooxGEPg8CrZgMfyvC4uj/djJa
vkIJ/Lnt4UxiNrl sO3WHkwt0wsqbys5k9Xo8
J7BmEHbSNSuzvdqdSCZNLqqTpXLAJ98wJ0WgZ
nG6UvZmddQTSDWKDHq Rh3bZSKxG8oQtwVc7G
XIVgm/stwj8E2e7IaYr9iVjXlVTiDxkxQOfvn
MeAOof2l2WXg8Q/x4SAtj Asbjip2qIhq2Dw4
8uHGP1ZoYalFEzTXwmO8ompUiWf5YB/Gt624X
A+iAcUiUZARWGSIR4lBaQGDi GwGGqY5Ud/oB
686Q/4y0hBvbUzfY5DKdBn/YlA==
```

```
Plain text output: This is a long
message!
```

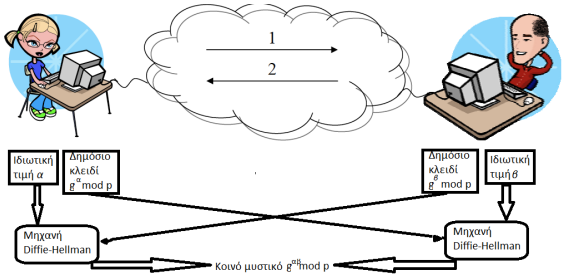
Παρατηρήστε ότι στην περίπτωση του αλγορίθμου El Gamal το μέγεθος του παραγόμενου κρυπτοκειμένου είναι διπλάσιο του μεγέθους του κλειδιού (1024).

Ο αλγόριθμος Diffie-Hellman

Ο αλγόριθμος Diffie-Hellman δουλεύει λίγο διαφορετικά. Δημιουργήθηκε για να επιλύσει το πρόβλημα που αναφέραμε αρχικά, δηλ. της συμφωνίας ενός κοινού κλειδιού μεταξύ δυο οντοτήτων. Έστω p ένας μεγάλος πρώτος ακέραιος αριθμός και g μια γεννήτρια του p.

1. Η Αλίκη επιλέγει ένα τυχαίο μεγάλο ακέραιο $\alpha \in [1, p-1)$, υπολογίζει το g^{α} mod p (το δημόσιο κλειδί της) και στέλνει το αποτέλεσμα στο Βασίλη.
2. Ο Βασίλης επιλέγει ένα τυχαίο μεγάλο ακέραιο $\beta \in [1, p-1)$, υπολογίζει το g^{β} mod p (το δημόσιο κλειδί του) και στέλνει το αποτέλεσμα στην Αλίκη.
3. Η Αλίκη υπολογίζει το k ← $(g^{\beta})^{\alpha}$ mod p
4. Ο Βασίλης υπολογίζει το k ← $(g^{\alpha})^{\beta}$ mod p

Το k είναι πλέον το κοινό μυστικό μεταξύ της Αλίκης και του Βασίλη. Π.χ., εφαρμόστε τον παραπάνω αλγόριθμο για α = 8, β = 37, p = 43, g = 3. Θα πρέπει να καταλήξετε στο k = 9.



Σχήμα 2: Λειτουργία του αλγορίθμου Diffie-Hellman.

Ένα παράδειγμα χρήσης του αλγορίθμου Diffie-Hellman φαίνεται στο πρόγραμμα SimpleASymmetricExampleBC που κατεβάσατε προηγουμένως, αν διώξετε το σχόλιο από την τελευταία γραμμή:

```
System.out.println("\nDiffie-Hellman
shared key: " + asymmetric.keyAgreement
("DH", "BC", asymmetric.generateKey
Pair("DH", "BC"), asymmetric.generate
KeyPair("DH", "BC")));
```

Η έξοδος του προγράμματος γίνεται πλέον:

```
Plain text input: This is a long
message!
Cipher text: CG/k4i/KJjArBfmWlbSt6TeL
v1jjDxOj6YqipMZGmxx1JtAFREVLHdy4kQJA
1zfKMS+5UZzKhM3K z+QOEax96hVYNw121V1
cdVWTH5CX8zMhYqZ2TjkH5JQ4oqIvmJfQ37ti
K/nPx6iPRaecsF2iX5OZ HToxADyVgEU3jn72
kTO=

Plain text output: This is a long
message!

Diffie-Hellman shared key:
WnUwvuovvDgmP2ClgtYSjiJ6G6wqQt68tPer
```

```
Dw72sEWLQ1l0fJOBlWFr4cSNKYWPIUWH6m64
GIXX zZm9qu4UOVIOAH0uZG61TzrWXTPVrrV
harL8wYzysiQWewhrcCOgLK+DMH2UYHPvvqV
VCv3qfa78 Dpui5Cpfl6F5SGXHoJU=
```

Παρατηρήστε ότι η μέθοδος keyAgreement() ακολουθεί το σχήμα 2, δηλ.

```
aKeyAgreement.doPhase(bKeyPair.get
Public(), true);
bKeyAgreement.doPhase(aKeyPair.get
Public(), true);
```

Παρόλ' αυτά, ο αλγόριθμος DH είναι ευάλωτος σε επιθέσεις ενδιάμεσων (man-in-the-middle attacks):

1. Η Αλίκη επιλέγει ένα τυχαίο μεγάλο ακέραιο α ∈R[1, p-1), υπολογίζει το g^{α} mod p (το δημόσιο κλειδί της) και στέλνει το αποτέλεσμα στην Εύα («Βασίλη»).

1'. Η Εύα («Αλίκη») υπολογίζει το $g_{\epsilon} = g^{\epsilon}$ mod p για κάποιο ε∈R[1, p-1) και το στέλνει στο Βασίλη.

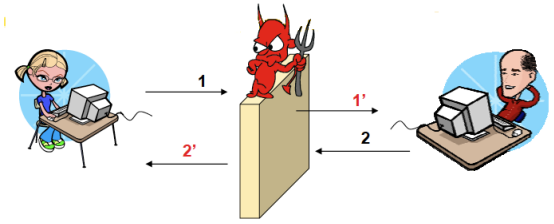
2. Ο Βασίλης επιλέγει ένα τυχαίο μεγάλο ακέραιο β∈[1, p-1), υπολογίζει το g^{β} mod p (το δημόσιο κλειδί του) και στέλνει το αποτέλεσμα στην Εύα («Αλίκη»).

2'. Η Εύα («Βασίλης») στέλνει το g_{ϵ} στην Αλίκη

3. Η Αλίκη υπολογίζει το k1 ← $(g^{\epsilon})^{\alpha}$ mod p (κοινό μυστικό μεταξύ αυτής και της Εύας)

4. Ο Βασίλης υπολογίζει το k2 ← $(g^{\epsilon})^{\beta}$ mod p (κοινό μυστικό μεταξύ του και της Εύας)

Στο επόμενο τεύχος θα δούμε πως μπορούμε ν' αντιμετωπίσουμε τέτοιου είδους επιθέσεις.



Σχήμα 3: Man-in-the-middle attack στον αλγορίθμο Diffie-Hellman.

Επίλογος

Αφού λοιπόν η κρυπτογραφία δημόσιου κλειδιού έχει όλα αυτά τα πλεονεκτήματα, γιατί δεν αντικατέστησε τη συμμετρική κρυπτογραφία; Ο λόγος είναι ότι οι αλγόριθμοι ασύμμετρης κρυπτογραφίας είναι πολύ πιο αργοί απ' της συμμετρικής κρυπτογραφίας. Έτσι, οι δυο αυτές κατηγορίες αλγορίθμων συνήθως χρησιμοποιούνται συμπληρωματικά. Ένα παράδειγμα προγράμματος που χρησιμοποιεί την κρυπτογραφία δημόσιου κλειδιού για να κρυπτογραφήσει το συμμετρικό κλειδί που θα αποσταλεί στον παραλήπτη ώστε να χρησιμοποιηθεί για τη μετέπειτα κρυπτογράφηση του κειμένου που όντως μας ενδιαφέρει μπορείτε να κατεβάσετε από `http://files.ubuntu-gr.org/ubuntistas/files/java_cryptography.zip`. Το πρόγραμμα δημιουργεί ένα συμμετρικό κλειδί και ένα ζεύγος δημόσιου/ιδιωτικού κλειδιού. Στη συνέχεια κρυπτογραφεί το συμμετρικό κλειδί χρησιμοποιώντας το δημόσιο κλειδί στην byte[] encrypt(final byte[] message, final Cipher cipher, final Key key), η οποία έχει τροποποιηθεί κατάλληλα ώστε να παράγει και συμμετρικά και ασύμμετρα κρυπτογραφημένο κείμενο, ενώ για απλότητα δεν χρησιμοποιείται η Base64. Στη συνέχεια, το κείμενό μας κρυπτογραφείται με χρήση του συμμετρικού κλειδιού, πάλι με την encrypt(). Κατά την αποκρυπτογράφηση, ανακτούμε πρώτα το συμμετρικό κλειδί, χρησιμοποιώντας το ιδιωτικό κλειδί στην byte[] decrypt(final byte[] cipherText, final Cipher cipher, final Key key) και περνώντας το αποτέλεσμα στην SecretKeySpec, καθώς και τον αλγόριθμο που χρησιμοποιήσαμε για ν' ανακτήσουμε το συμμετρικό κλειδί. Τέλος, αποκρυπτογραφούμε το μήνυμα χρησιμοποιώντας το ανακτημένο συμμετρικό κλειδί. Ένα παράδειγμα εκτέλεσης του προγράμματος φαίνεται παρακάτω:

```
$ java -cp .:lib/bcprov-jdk16-145.jar
gr.ubuntistas.issue10.CombinedSymme
tricASymmetricExample This is a long
message!
Plain text input: This is a long
message!
Encrypted secret key: [B@2d58f9d3
Cipher text: [B@175093f1
Recovered secret key: [B@72e6f7d2
Plain text output: This is a long
message!
```

Ο παρακάτω πίνακας παρουσιάζει το ελάχιστο μέγεθος του κλειδιού που θα πρέπει να χρησιμοποιείται στο μέλλον, αν

θεωρήσουμε ότι θα ισχύει ο νόμος του Moore για την πρόοδο της κρυπτανάλυσης.

'Ετος	AES	RSA, DH, ElGamal
2010	78	1369
2020	86	1881
2030	93	2439
2040	101	3214

Πηγές:

1. Menezes A., van Oorschot P., Vanstone S. (1997), Handbook of Applied Cryptography, CRC Press.

2. Schneier B. (1996), Applied Cryptography, John Wiley & Sons.

3. Anderson R. (2001), Security Engineering, John Wiley & Sons.

4. Stallings W. (2005), Cryptography and Network Security, Prentice Hall.

5. Hook D. (2005), Beginning Cryptography with Java, Wrox.

6. Knudsen J. (1998), Java Cryptography, O' Reilly.

7. Hanna T. (2010), “Methods of Secrecy”, Hakin9, τεύχος 2, σελ. 54-57.

8. Hanna T. (2010), “Symmetric Secrets”, Hakin9, τεύχος 3, σελ. 50-54.

9. Schratt M. (2009), “RSA & AES in JAVA”, Hakin9, τεύχος 5, σελ. 52-57.

10. http://el.wikipedia.org/wiki/Κρυπτογράφηση_Δημόσιου_Κλειδιού

11. Java Tutorial, Security, `http://www.java2s.com/Tutorial/Java/0490__Security/Catalog0490__Security.htm`.

(Συνέχεια από τη σελίδα 9)

Επίλογος

Συμπερασματικά, το Mercurial διακρίνεται για την αμεσότητα και την ευκολία χρήσης και υπερτερεί σε σχέση με άλλα συστήματα διαχείρισης εκδόσεων τόσο σε απόδοση όσο και σε επεκτασιμότητα. Δεν χρειάζεται συντήρηση, ούτε διαχειριστή, και δουλεύει καλά με κάθε είδους προγραμματιστές και κάθε είδους έργα. Ίσως σε κάποιο επόμενο τεύχος να δώσουμε κάποιες οδηγίες για το πώς μπορείτε να δημοσιεύσετε ένα repository σε κάποιον web server όπως ο lighttpd ή ο Apache, ώστε να είναι διαθέσιμο στον παγκόσμιο ιστό. Ως τότε, σας ευχόμαστε καλή διαχείριση των εκδόσεων των αρχείων σας με το Mercurial. :)

Πηγές:

1. O’Sullivan B. (2009), Mercurial: The Definitive Guide, O' Reilly.

2. Mercurial wiki, `http://mercurial.selenic.com/wiki/`

του Πούλιου Κωνσταντίνου

Ελεύθερο Υλικό

Στα χνάρια του ελεύθερου λογισμικού!

Οι περισσότεροι αναγνώστες του Ubuntistas είναι εξοικειωμένοι με την έννοια του ελεύθερου λογισμικού. Το παρόν άρθρο, όμως, έχει ως θέμα του κάτι λιγότερο διαδεδομένο, το ελεύθερο υλικό, το οποίο θα εξερευνήσουμε μέσα από το παράδειγμα των τρισδιάστατων εκτυπωτών ανοιχτού "κώδικα". Παράλληλα, επιχειρείται μια σύγκριση μεταξύ των συνθηκών ανάπτυξης ελεύθερου λογισμικού κι ελεύθερου υλικού.

Ως ελεύθερο υλικό μπορεί να ορισθεί το σύνολο των συσκευών ή κατασκευών των οποίων τα κατασκευαστικά σχέδια, οι λίστες τεμαχίων και όλες οι απαραίτητες οδηγίες κατασκευής και συναρμολόγησης είναι στην διάθεση του καθενός με σκοπό την απεριόριστη παραγωγή τους. Συνήθως, όταν κανείς αναφέρεται σε ελεύθερο υλικό, το πρώτο πράγμα που έρχεται στο νου είναι ηλεκτρονικές διατάξεις, στην πράξη όμως ο όρος μπορεί να αναφέρεται σε οποιαδήποτε συσκευή, από μανταλάκια μέχρι οχήματα, ή ακόμη και μεγάλες κατασκευές, όπως γέφυρες.

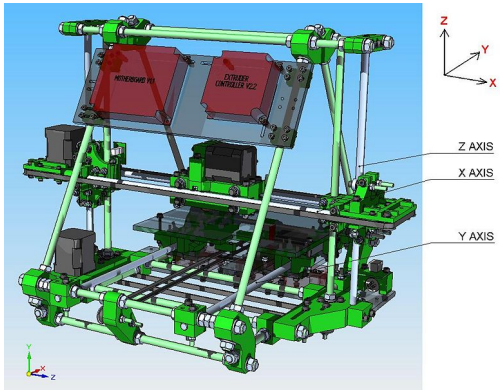
Τρισδιάστατοι εκτυπωτές / Το έργο RepRap

Ένας τρισδιάστατος εκτυπωτής αποτελεί μία ενδιαφέρουσα περίπτωση υλικού, το οποίο συνδυάζει τόσο ηλεκτρονικά όσο και μηχανολογικά μέρη. Πρόκειται για μία συσκευή η οποία φέρει μία οδηγούμενη κεφαλή όπως ένας κανονικός εκτυπωτής, αλλά αντί για μελάνι εναποθέτει κάποιο είδος πλαστικού, το οποίο στερεοποιείται καθώς ψύχεται. Με την κίνηση της κεφαλής να γίνεται σε τρεις διαστάσεις μπορούν να παραχθούν πλαστικά αντικείμενα οποιασδήποτε μορφής. Μία επιπλέον διαφορά σε σχέση με τους κοινούς εκτυπωτές χαρτιού είναι το υψηλό κόστος αγοράς, που στην περίπτωση κλειστών εμπορικών λύσεων [1] ξεκινά περίπου στα 10.000 €.

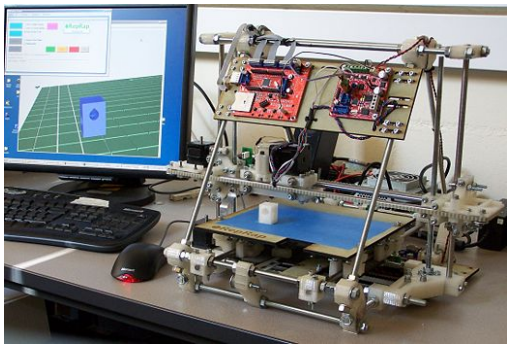
Ένα πολύ αξιόλογο έργο ανάπτυξης ενός τρισδιάστατου εκτυπωτή ανοιχτού "κώδικα" είναι το RepRap [2] του οποίου ο βασικός στόχος είναι η δημιουργία μιας συσκευής που να μπορεί να αναπαράγει τον εαυτό της. Στην περίπτωση αυτή, το κόστος των εξαρτημάτων για την κατασκευή του τρισδιάστατου εκτυπωτή ανέρχεται σε μόλις 500 €. Φυσικά, θα πρέπει κανείς να συνυπολογίσει ότι ο χρόνος που απαιτείται για να συναρμολογηθεί και να τεθεί σε λειτουργία η συσκευή δεν είναι καθόλου αμελητέος.

Το έργο RepRap έχει δημιουργήσει μέχρι στιγμής τρία διαφορετικά μοντέλα εκτυπωτών, δίνοντάς τους, αντίστοιχα, τα ονόματα των διάσημων βιολόγων Darwin, Mendel και Huxley. Το μοντέλο Mendel είναι το κύριο μοντέλο του έργου και αποτελεί ανασχεδιασμό και επέκταση του αρχικού μοντέλου Darwin. Το μοντέλο Huxley είναι μια πρόσφατη προσπάθεια για έναν εκτυπωτή ελαφρώς μικρότερων διαστάσεων.

Στις εικόνες 1 και 2 παρουσιάζεται το μοντέλο Mendel. Τα τμήματα της πρώτης εικόνας με πράσινο χρώμα μπορούν να κατασκευαστούν από τον ίδιο τον εκτυπωτή, εάν κάποιος θελήσει να κατασκευάσει ένα δεύτερο, τρίτο κτλ. αντίγραφο. Η προμήθεια τμημάτων όπως οι μεταλλικές ράβδοι, τα έδρανα κυλίσεως και οι βηματικοί κινητήρες θα πρέπει να γίνει ξεχωριστά, με κόστος περί τα 500 € για κάθε αντίγραφο, ενώ τα ηλεκτρονικά τμήματα του εκτυπωτή βασίζονται στην πλακέτα Arduino που είναι επίσης ανοιχτού κώδικα [3].

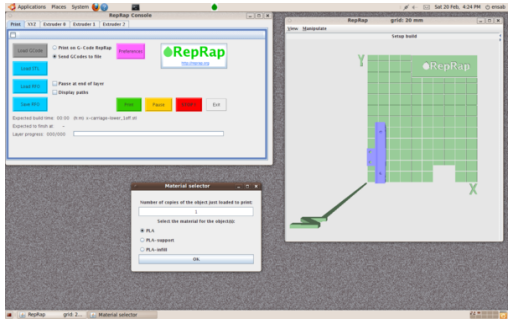


Εικόνα 1: Τρισδιάστατη CAD αναπαράσταση του μοντέλου Mendel.



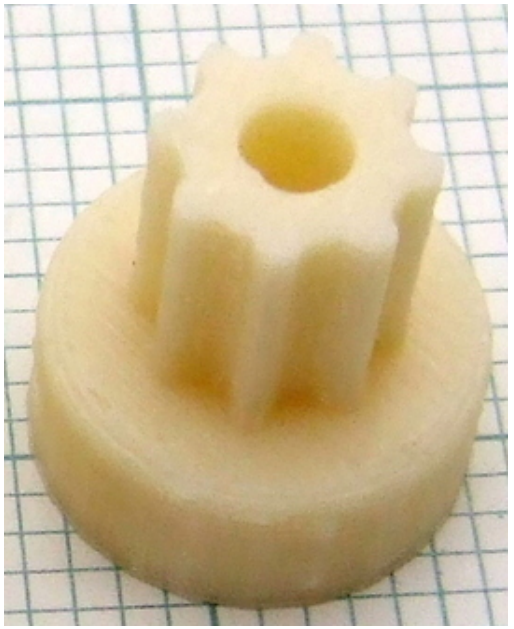
Εικόνα 2: Το μοντέλο Mendel εν δράσει.

Όσον αφορά το λογισμικό χειρισμού του εκτυπωτή, πρόκειται φυσικά για ελεύθερο λογισμικό [4], ένα στιγμιότυπο του οποίου παρουσιάζεται στην εικόνα 3. Το λογισμικό είναι σε θέση να διαβάζει αρχεία της μορφής STL και να τα μετατρέπει σε κώδικα μηχανής G, βάσει του οποίου λειτουργεί ο εκτυπωτής. Η δημιουργία των μοντέλων σε μορφή STL μπορεί να γίνει π.χ. με το πολύ διαδεδομένο ελεύθερο λογισμικό Blender, ή με εμπορικά προγράμματα όπως το Solid-Works. Πρακτικά, όλα τα προγράμματα τρισδιάστατης σχεδίασης, ελεύθερα η μη, μπορούν να εξάγουν αρχεία της μορφής STL.



Εικόνα 3: Το λογισμικό χειρισμού του εκτυπωτή.

Η εικόνα 4 δίνει ένα παράδειγμα αντικειμένου που μπορεί να κατασκευάσει ένας τρισδιάστατος εκτυπωτής του έργου RepRap. Η ακρίβεια της κατεργασίας είναι αρκετά περιορισμένη κι εξαρτάται από το υλικό που χρησιμοποιείται, αλλά και από την εμπειρία του χειριστή στο να κάνει τροποποιήσεις και βελτιστοποιήσεις στον κώδικα μηχανής για την παραγωγή του τεμαχίου. Ωστόσο το μεγάλο πλεονέκτημα της συσκευής είναι οι πάρα πολλές δυνατότητες τροποποιήσεων κι επέκτασής της. Ήδη έχουν δημιουργηθεί παραλλαγές της που μπορούν να τυπώνουν ηλεκτρονικά κυκλώματα, ή να κάνουν κατεργασία που περιλαμβάνει αφαίρεση υλικού, χρησιμοποιώντας κεφαλή παρόμοια με αυτήν μιας φρέζας, αλλά με πολύ μικρότερες διαστάσεις. Το σίγουρο είναι ότι όσοι ασχοληθούν με την κατασκευή και τον χειρισμό/προγραμματισμό μιας τέτοιας μηχανής θα αποκτήσουν πάρα πολύ χρήσιμες γνώσεις στα πεδία του αυτοματισμού και των κατεργασιών.



Εικόνα 4: Τροχαλία κατασκευασμένη μέσω εκτύπωσης 3Δ.

Παραλληλισμοί, συγκρίσεις και συμπεράσματα

Στην συνέχεια αναζητούμε κοινά σημεία και διαφορές μεταξύ των κοινοτήτων ελεύθερου λογισμικού και ελεύθερου υλικού. Ο πίνακας 1 συγκεντρώνει κάποιες αντιστοιχίες μεταξύ της ανάπτυξης λογισμικού και υλικού.

Λογισμικό	Υλικό
Ψηφιακές ανάγκες π.χ. αποθήκευση, οργάνωση, αναπαράσταση, διαχείριση δεδομένων, σύνθετοι υπολογισμοί, επικοινωνίες	Υλικές ανάγκες π.χ. ηλεκτρικές, ηλεκτρονικές, μηχανολογικές συσκευές, έργα υποδομών, μεταφορές
Προγράμματα Η/Υ π.χ. συστήματα βάσεων δεδομένων, εφαρμογές σχεδιασμού και προσομοίωσης, εφαρμογές και πρωτόκολλα επικοινωνίας	Φυσικά προϊόντα π.χ. ψυγεία, τηλεοράσεις, ιατρικός εξοπλισμός, οχήματα, κατοικίες, δρόμοι, γέφυρες, σιδηρόδρομοι
Μεταγλώττιση (Compiling, Linking) π.χ. gcc, g++, gfortran, python interpreter, java runtime engine, ld, automake, cmake	Διαδικασία παραγωγής π.χ. χύτευση, αφαίρεση υλικού, συγκόλληση, πλαστική παραμόρφωση, χημική προσβολή, θερμική κατεργασία, εκσκαφή, οδόστρωση, συναρμολόγηση
Πηγαίος κώδικας π.χ. C, C++, Fortran, Python	Κατασκευαστικά σχέδια, λίστες τεμαχίων, παράμετροι κατεργασίας, οδηγίες συναρμολόγησης
Εξοπλισμός: π.χ. Η/Υ, σύνδεση ίντερνετ	Εξοπλισμός, πρώτες ύλες: π.χ. διαθέσιμος χώρος, τόρνοι, φρέζες, ρομπότ, οδοστρωτήρες, γερανοί, ορυκτές ύλες, χάλυβας, λιπαντικά
Προγραμματιστές, γραφίστες, χρήστες	Μηχανικοί, τεχνίτες, χρήστες

Πίνακας 1: Αντιστοιχίες μεταξύ λογισμικού και υλικού

Αναφερόμενοι στην κοινότητα ανάπτυξης ελεύθερου λογισμικού εννοούμε συνήθως ένα σύνολο προγραμματιστών και χρηστών που, είτε εθελοντικά είτε ως εργαζόμενοι σε εταιρείες του χώρου, συνεργάζονται για την ανάπτυξη των προγραμμάτων που χρησιμοποιούμε καθημερινά. Ακόμα και για αυτούς που η συμμετοχή τους είναι επαγγελματική, στις περισσότερες περιπτώσεις παραμένει θέμα αρχής το προϊόν της εργασίας τους να είναι ανοιχτό κι ελεύθερο. Η πλειοψηφία τους δεν θα δεχόταν μία θέση σε κάποια εταιρεία κλειστού λογισμικού ακόμα κι αν τους εξασφάλιζε έναν υψηλότερο μισθό. Αντίστοιχα μία κοινότητα ανάπτυξης ελεύθερου υλικού αποτελείται από μηχανικούς, θεωρητικούς επιστήμονες, τεχνίτες και χρήστες οι οποίοι συνεργάζονται για την ελεύθερη σχεδίαση και παραγωγή ενός προϊόντος. Προς το παρόν η κοινότητα ελεύθερου υλικού είναι εξαιρετικά περιορισμένη ενώ, πέρα από τον τομέα των ηλεκτρονικών κυκλωμάτων, υπάρχουν ελάχιστες επιχειρήσεις που δραστηριοποιούνται σε αυτόν τον χώρο. Μία βασική διαφορά μεταξύ της ανάπτυξης λογισμικού και της ανάπτυξης υλικού είναι ο αναγκαίος εξοπλισμός και πρώτες ύλες. Στην περίπτωση του λογισμικού αρκεί ένας Η/Υ και μία σύνδεση στο ίντερνετ. Στην περίπτωση του υλικού όμως, απαιτούνται εργαλεία και πρώτες ύλες που μπορεί να έχουν κάποιο σημαντικό κόστος. Αυτή η διαφορά θα μπορούσε να αιτιολογήσει σε κάποιον βαθμό την τόσο περιορισμένη ανάπτυξη της κοινότητας ελεύθερου υλικού. Όμως μία λίγο προσεκτικότερη προσέγγιση θα μας φανερώσει μια πιθανώς ουσιαστικότερη διαφορά. Στην κοινότητα ελεύθερου λογισμικού πρωταγωνιστής είναι ο προγραμματιστής, ενώ στην κοινότητα ελεύθερου υλικού αυτόν τον ρόλο καλείται να τον αναλάβει ο μηχανικός. Μηχανικοί και προγραμματιστές έχουν πολλές ομοιότητες, αλλά διαφέρουν σημαντικά στον βαθμό εξειδίκευσης. Ο μέσος προγραμματιστής γνωρίζει καλά τουλάχιστον 4-5 γλώσσες προγραμματισμού και, αν διαθέσει τον απαιτούμενο αριθμό εργατοωρών, είναι σε θέση να δημιουργήσει μόνος του

προγράμματα υψηλής πολυπλοκότητας και σημαντικού όγκου. Φυσικά προτιμάει να δουλεύει σε μεγαλύτερες ομάδες συνεργασίας, αλλά διαθέτει την αυτοπεποίθηση ότι ακόμη και μόνος του μπορεί να δημιουργήσει ένα ολοκληρωμένο προϊόν. Αντιθέτως, ο μέσος μηχανικός σήμερα στερείται αυτής της αυτοπεποίθησης. Αυτή η αδυναμία του μεμονωμένου μηχανικού να οραματιστεί την ανεξάρτητη δημιουργία ενός ολοκληρωμένου προϊόντος οφείλεται εν μέρει στο υψηλό κόστος σε υποδομές και πρώτες ύλες που είναι απαραίτητες. Επιπλέον όμως, ο μέσος μηχανικός έχει συνηθίσει να ασχολείται με κάτι αρκετά εξειδικευμένο, χωρίς να έχει την εποπτεία του συνόλου. Είναι εξαρτημένος από την ομάδα στην οποία εργάζεται, η οποία συχνά αριθμεί δεκάδες ή εκατοντάδες μηχανικούς. Ποιο θα είναι το μέλλον της κοινότητας ελεύθερου υλικού; Είναι πιθανόν, από την στιγμή που θα δημιουργηθεί ένας αρχικός πυρήνας και υπάρξουν κάποια πρώτα επιτυχημένα έργα, να παρατηρηθεί η ίδια εκθετική ανάπτυξη που γνώρισε και η κοινότητα ελεύθερου λογισμικού τα τελευταία 20 χρόνια. Στην ιδανική μάλιστα περίπτωση, η κοινότητα ελεύθερου υλικού θα μπορούσε να συντελέσει σε μία νέα βιομηχανική επανάσταση. Μία αρκετά πλήρης λίστα με έργα ελεύθερου υλικού δίνεται στον σύνδεσμο [5]. Για περισσότερες πληροφορίες, μπορείτε επίσης να διαβάστε το άρθρο του συνδέσμου [6].

Βιβλιογραφία

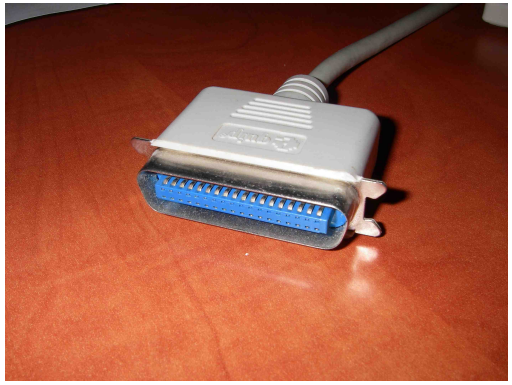
1. http://www.dimensionprinting.com/3d-printers/3d-printing-uprint-video.aspx
2. http://reprap.org
3. http://el.wikipedia.org/wiki/Arduino
4. http://sourceforge.net/projects/reprap/
5. http://p2pfoundation.net/Product_Hacking
6. http://www.we-magazine.net/we-volume-02/the-emergence-of-open-design-and-open-manufacturing/

του the_eye

Αυτοματισμοί με την παράλληλη θύρα

και όχι μόνο...

Η παράλληλη θύρα είναι μια από τις πιο πολυχρησιμοποιημένες θύρες. Στο παρελθόν ήταν σχεδόν μονόδρομος για τους εκτυπωτές. Η σύνδεσή της, τύπου D25 ή cetronics, είναι πολύ εύκολα αναγνωρίσιμη σε πολλούς από εμάς.



Εικόνα 1.

Η χρήση της υπήρξε τόσο ευρεία χάρη στην απλότητα του πρωτοκόλλου. Η παράλληλη θύρα χρησιμοποιεί παράλληλη μετάδοση 8 bits σε κάθε διακριτό χρονικό διάστημα. Μπορεί να μεταδώσει ένα byte κάθε χρονική στιγμή. Ακόμα, το πρωτόκολλό της είναι πολύ ανεκτικό σε καθυστερήσεις και λάθη, με αποτέλεσμα να είναι ιδανική για πειραματισμό. Σε αυτό το άρθρο θα παρουσιάσουμε κάποιους αυτοματισμούς που μπορούν να γίνουν χρησιμοποιώντας την θύρα αυτή. Έχουμε δει σε εκθέσεις ή σε τηλεοπτικές εκπομπές τα λεγόμενα “έξυπνα σπίτια”, τα οποία μπορούμε να προγραμματίσουμε ώστε να εκτελούν κάποιες επιθυμητές ενέργειες. πχ να ανάβουν τα φώτα ή να ανοιγοκλείνουν το θερμοσίφωνα αυτόματα. Δεν θα επεκταθούμε τόσο πολύ στο παρόν άρθρο, αλλά θα φτάσουμε αρκετά κοντά. Ας δούμε τη θύρα στα ενδότερά της. Η σύνδεση D25 έχει 25 επαφές, που έχουν την παρακάτω διάταξη και αρίθμηση:

13 12 11 10 9 8 7 6 5 4 3 2 1
25 24 23 22 21 20 19 18 17 16 15 14

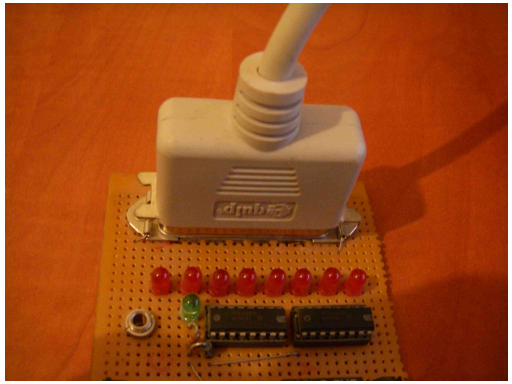
Η χρήση των επαφών που μας ενδιαφέρουν έχει ως εξής :

Επαφή:	Χρήση
2:	Bit1
3:	Bit2
4:	Bit3
5:	Bit4
6:	Bit5
7:	Bit6
8:	Bit7
9:	Bit8

Οι υπόλοιπες επαφές είναι για σηματοδοσία ή για έλεγχο, ενώ αρκετές, από την 18 μέχρι την 25, είναι γειώσεις. Όπως είπαμε και παραπάνω, το πρωτόκολλο είναι πολύ ανεκτικό, επομένως μπορούμε να τις παραλείψουμε. Το bit, όπως γνωρίζετε, είναι η μικρότερη μονάδα πληροφορίας, το λογικό 0 ή 1. Στην παράλληλη θύρα, και συγκεκριμένα στις επαφές 2 έως 9, το bit 0 αντιστοιχεί σε τάση ρεύματος 0 Volt και το bit 1 σε τάση 5 Volt . Παρακάτω βλέπετε τι αποτελέσματα προκύπτουν αν στείλουμε στην παράλληλη θύρα ορισμένες ενδεικτικές τιμές:

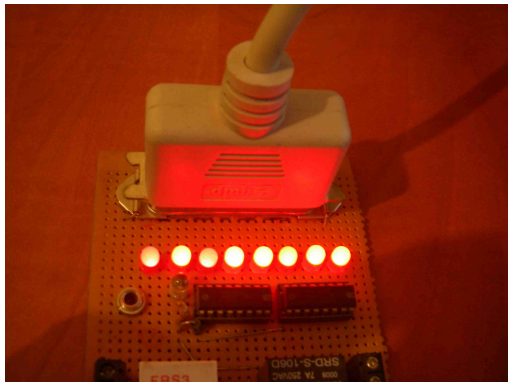
Δεκαδικό	Δεκαεξαδικό	Δυαδικό	Αποτέλεσμα	Εικόνα
0	00	00000000	Δεν εμφανίζεται πουθενά τάση 5V	Εικ. 2
255	FF	11111111	Εμφανίζεται τάση 5V σε όλες τις επαφές, από την 2 έως την 9.	Εικ. 3
170	AA	10101010	Εμφανίζεται τάση εναλλάξ.	Εικ. 4

Για να στείλουμε τα παραπάνω bytes στη θύρα, θα χρειαστεί να γράψουμε το παρακάτω πρόγραμμα σε C και να το μεταγλωττίσουμε με έναν compiler, τον gcc, που είναι διαθέσιμος σε κάθε σύγχρονο λειτουργικό σύστημα GNU/Linux.



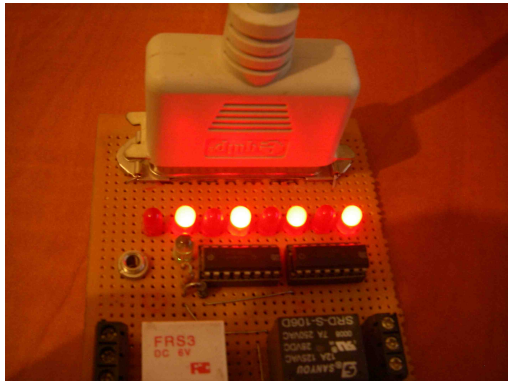
Εικόνα 2.

Ο κώδικας είναι πολύ απλός και το μόνο που κάνει είναι να δέχεται έναν αριθμό σαν παράμετρο και να τον αποθηκεύει στη βασική θέση μνήμης της θύρας.



Εικόνα 3.

Η βασική θέση μνήμης είναι στο 0x03bc για το /dev/lp0, στο 0x0378 για το /dev/lp1, και στο 0x0278 για το /dev/lp2. Στο δικό μου PC είναι στο 0x0378.



Εικόνα 4.

Υπάρχει τρόπος και για να διαβάζουμε δεδομένα από την παράλληλη θύρα, αλλά δεν θα επεκταθούμε ως εκεί στο παρόν άρθρο. Ακολουθεί ο κώδικας του αρχείου lp.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>

/* Basic memory position */
#define base 0x0378

main(int argc, char** argv)
{
int value;
sscanf(argv[1], "%d", &value);

   if (ioperm(base,1,1))
    fprintf(stderr, "Couldn't get the
       port at %x\n", base), exit(1);

   outb(value, base);
}
```

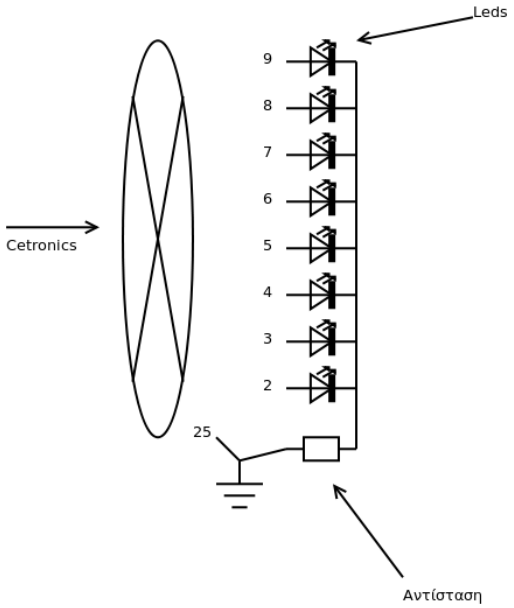
Για να μεταγλωττίσουμε τον κώδικα, τρέχουμε σε τερματικό:
gcc lp.c -o lp
Παράγεται το αρχείο lp, το οποίο πρέπει να έχει δικαιώματα εκτέλεσης rwxr-xr-x. Η κατάλληλη ρύθμιση των δικαιωμάτων γίνεται με την εντολή chmod.

Αν όλα έχουν πάει καλά, θα έχουμε αποκτήσει μια εφαρμογή με την οποία θα μπορούμε να ελέγξουμε τα 8 bits της παράλληλης θύρας. Εύκολα τα πράγματα ως εδώ!

Τρέχουμε την εφαρμογή με sudo, γιατί μόνο ο υπερχρήστης έχει δικαίωμα άμεσης επέμβασης στο υλικό, και πληκτρολογούμε π.χ. sudo ./lp 170

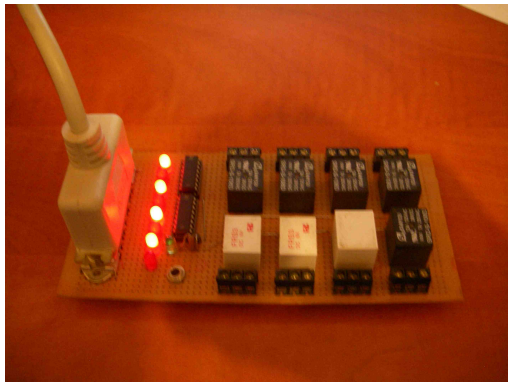
Θα πείτε όλα καλά, δεν έβγαλε κανένα μήνυμα σφάλματος, αλλά τι σχέση έχει αυτό με το ηλεκτρονικό σούπερ σπίτι που αναφέραμε στην αρχή; Έχει!

Πάμε στο δεύτερο τμήμα της παρουσίασης, που ασχολείται με βασικά ηλεκτρονικά (μην τρομάζετε). Υλικά: 1 καλώδιο παράλληλης σύνδεσης από εκτυπωτή (cetronics), 1 σύνδεση cetronics για πλακέτα, 8 leds (λαμπάκια), 1 αντίσταση 470 Ω και, προαιρετικά, μια διάτρητη πλακέτα βακελίτη. Το κύκλωμα διαμορφώνεται ως εξής:



Εικόνα 5.

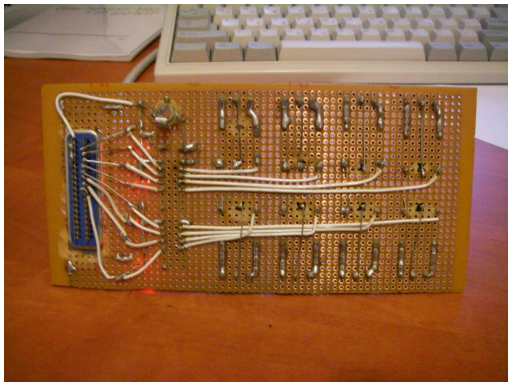
Προσοχή στην πολικότητα των leds. Αν δεν ανάβουν, πιθανώς τα έχετε τοποθετήσει ανάποδα. Σε αυτό το στάδιο της κατασκευής, όποια τιμή στέλνουμε με το πρόγραμμα ως δεκαδικό αριθμό, τη βλέπουμε να εμφανίζεται στα leds ως δυαδικός. Για να ολοκληρώσουμε την κατασκευή θα πρέπει να συνδέσουμε τα παραγόμενα σήματα με κάποιους διακόπτες ρελέ, αφού πρώτα τα ενισχύσουμε.



Εικόνα 6.

Για την ενίσχυση των σημάτων μπορούν να χρησιμοποιηθούν τρανζίστορ ή darlington gates ή άλλες ηλεκτρονικές διατάξεις. Κατόπιν, τα σήματα θα ελέγξουν 8 ρελέ των 220 Volt, και θα μας δώσουν 8 πραγματικούς διακόπτες ON/OFF, που με μια εντολή μας από την κονσόλα θα μπορούν να ανοίγουν ή να κλείνουν 8 διαφορετικές συσκευές, αντίστοιχα. Μπορούμε να συνδυάσουμε τον κώδικά μας με κώδικα PHP και να δημιουργήσουμε μια διαδικτυακή εφαρμογή για να ελέγχουμε απομακρυσμένα τους διακόπτες/συσκευές. Δείτε για παράδειγμα τη δική μου: http://ntoulasd.selfip.com:83/webremote (ελπίζω να μην μου κλείσετε το κλιματιστικό ;-)

Οι δυνατότητες για πειραματισμό είναι ανεξάντλητες. Μπορείτε να ελέγξετε κινητήρες, φώτα, ηλεκτροβάνες, είτε από κονσόλα, είτε μέσω του Cron, είτε από οπουδήποτε αλλού μπορεί να εκτελεστεί το πρόγραμμα.



Εικόνα 7.

του Παπαδόπουλου Δημήτρη

pyserial

Έλεγχος της παράλληλης και της σειριακής θύρας από python.

Η παράλληλη (παλιότερα χρησιμοποιούταν στους εκτυπωτές) και η σειριακή (παλιότερα συνδεόταν το modem) θύρα είναι ίσως τεχνολογικά ξεπερασμένες σήμερα, αλλά παρόλ' αυτά συνεχίζουν να χρησιμοποιούνται αρκετά στη βιομηχανία. Με την παράλληλη θύρα μπορεί κανείς να ελέγξει απευθείας μικροδιακόπτες (relais) και, κατά συνέπεια, διάφορες ηλεκτρικές και ηλεκτρονικές συσκευές. Όποιος έχει τις γνώσεις μπορεί να κάνει τις κολλήσεις μόνος του, διαφορετικά υπάρχουν έτοιμες κάρτες ή προσαρμογείς. Η σειριακή θύρα χρησιμοποιείται για την μεταφορά σήματος σε διάφορες άλλες συσκευές.

Ο έλεγχος των θυρών αυτών γίνεται συνήθως με C, αλλά γιατί να μπλέκει κανείς με C, όταν μπορεί να χρησιμοποιήσει python με λιγότερο κόπο; Το πακέτο που θα χρησιμοποιήσουμε είναι το pyserial και μπορεί να βρεθεί εδώ: `http://pyserial.sourceforge.net/`. Η εγκατάσταση γίνεται όπως πάντα για πακέτα python, με την εντολή: `python setup.py install`

Μέσα από την python, το μόνο που πρέπει να κάνουμε για να το χρησιμοποιήσουμε είναι να το εισάγουμε με: `import serial`

Για τη σωστή χρήση του pyserial οφείλουμε να γνωρίζουμε τα τεχνικά χαρακτηριστικά της θύρας. Ας δούμε όμως ένα παράδειγμα για επικοινωνία με τη σειριακή θύρα. Ας υποθέσουμε ότι θέλουμε να γράψουμε στη θύρα 0 τους χαρακτήρες 'hello'. Τότε, γράφουμε σε python πολύ απλά:

```
import serial
ser = serial.Serial(0)
print ser.portstr
ser.write("hello")
ser.close()
```

Με την πρώτη εντολή φορτώνουμε το απαραίτητο module, ενώ με τη δεύτερη ανοίγουμε τη σειριακή θύρα 0 και επιστρέφουμε ένα αντικείμενο στη μεταβλητή ser. Έπειτα, τυπώνουμε το πραγματικό όνομα της θύρας που ανοίξαμε (σε κάποια λειτουργικά μπορεί να έχει διαφορετική ονομασία). Μετά, καλούμε μια από της μεθόδους του αντικειμένου serial, τη write, και γράφουμε εκεί τους χαρακτήρες που θέλουμε. Τέλος, κλείνουμε τη θύρα. Αν κρατήσουμε τη θύρα ανοιχτή, μπορεί να προκύψουν παράξενα και δύσκολα bugs, γι' αυτό συνιστούμε το κλείσιμό της αμέσως μετά τη χρήση της.

Φυσικά, η παραπάνω χρήση θεωρεί κάποιες τιμές ως προεπιλεγμένες. Για παράδειγμα, το baudrate, δηλαδή ο ρυθμός μετάδοσης πληροφορίας, είναι 9600, το parity είναι None, το μέγεθος των bits 8, και το πλήθος των stopbits 1. Οι τιμές αυτές, αλλά και ορισμένες άλλες, διαφέρουν από συσκευή σε συσκευή. Για τις λεπτομέρειες αυτές δείτε το τεχνικό εγχειρίδιο της συσκευής σας. Οι τιμές μπορούν να οριστούν είτε κατά το άνοιγμα της θύρας:

```
ser = serial.Serial(1, 38400, timeout=0,
... parity=serial.PARITY_EVEN, rtscts=1)
```

είτε αργότερα, τροποποιώντας τις ιδιότητες του αντικειμένου:

```
ser.baudrate = 19200
ser.stopbits = 1
```

Με παρόμοιο τρόπο μπορούμε να διαβάσουμε δεδομένα από τη σειριακή θύρα, χρησιμοποιώντας τις εντολές read ή readline:

```
s = ser.read(100)
```

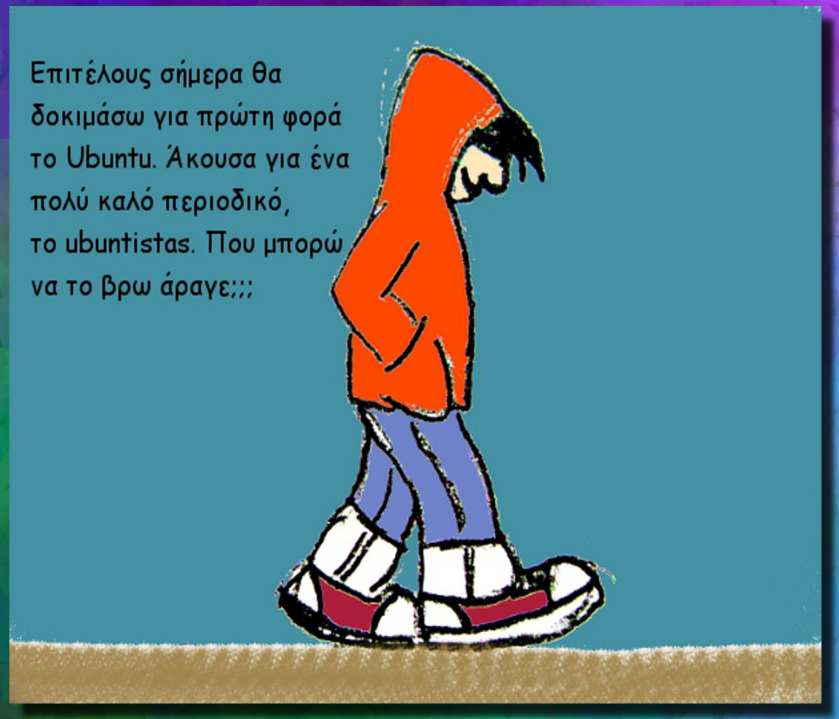
Η παραπάνω εντολή διαβάζει τα 100 bytes που βρίσκονται στο buffer, ενώ η

```
line = ser.readline()
```

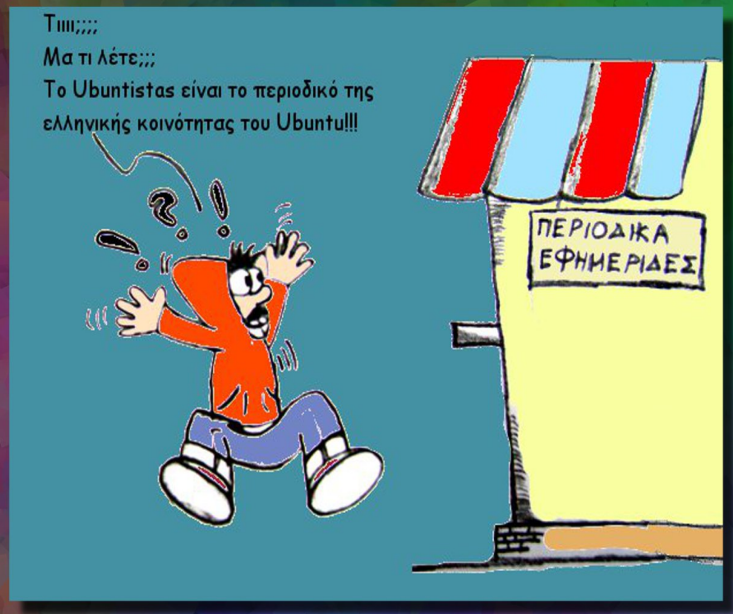
διαβάζει μέχρι να συναντήσει το χαρακτήρα '\n'. Όλα αυτά εξαρτώνται από τις τεχνικές προδιαγραφές της εκάστοτε συσκευής.

Όλες οι λεπτομέρειες για το API του pyserial μπορούν να βρεθούν στην ιστοσελίδα `http://pyserial.sourceforge.net/pyserial_api.html`. Με το πακέτο αυτό μπορείτε να ελέγξετε από απλές λάμπες ή καφετιέρες μέχρι ολόκληρες βιομηχανικές εγκαταστάσεις.


UBUNTιστορίες
by GREG
Heavily GIMPed


Επιτέλους σήμερα θα δοκιμάσω για πρώτη φορά το Ubuntu. Άκουσα για ένα πολύ καλό περιοδικό, το ubuntistas. Που μπορώ να το βρω άραγε;;;


Ας ρωτήσω στο περίπτερο ...
Γεια σας! Μήπως έχετε το Ubuntistas;
Δεν έχουμε ισπανικά περιοδικά κύριε...!
ΠΕΡΙΟΔΙΚΑ ΕΦΗΜΕΡΙΔΕΣ


Τιιι;;;;
Μα τι λέτε;;;
Το Ubuntistas είναι το περιοδικό της ελληνικής κοινότητας του Ubuntu!!!
ΠΕΡΙΟΔΙΚΑ ΕΦΗΜΕΡΙΔΕΣ


Αααα, ναι, ναι... θυμήθηκα!
Μα είστε εντελώς άσχετος αγαπητέ μου!
Το ubuntistas είναι ηλεκτρονικό περιοδικό, το οποίο μπορείς να κατεβάσεις πολύ εύκολα από τη διεύθυνση http://ubuntistas.ubuntu-gr.org/
Θα μπορούσα να σας το κατεβάσω, αλλά δυστυχώς έχει κολλήσει((ο υπολογιστής μου...
Μάλλον πρέπει να βάλω κι εγώ το ubuntu τελικά...
Μάλλον...
ΠΕΡΙΟΔΙΚΑ ΕΦΗΜΕΡΙΔΕΣ