Ubuntistas

Το περιοδικό της Ελληνικής Κοινότητας Ubuntu-gr

Τεύχος #15
Οκτώβριος 2012

FF Multi Converter
Torcs
Open Source Ecology
Kiku & αναγνώριση φωνής
Foremost & επαναφορά δεδομένων

2ο Μέρος

LibreOffice

OS Performance

Κρυπτογράφηση με Vault

Erlang

Ομάδα Περιοδικού

Συντονιστές

Γιώργος Χριστοφής (Geochr)
[geochr22@gmail.com]
Φίλιππος-Χρήστος Φωτιάδης (filippos.xf)
[peacefulwarrior7891@gmail.com]

Συντάκτες

Γιάννης Κωστάρας (hawk)
[jkost@freemail.gr]
Κωνσταντίνος Αποστόλου (conmarap)
[conmarap@osarena.net]
Γιώργος Χριστοφής (Geochr)
[geochr22@gmail.com]
Νίκος Βασιλειάδης (Wolfsoul)
[vasileiadis.nikos@gmail.com]
Γιώργος Μακρής (ubuderix)
[geo_mak200@yahoo.com]
Γιάννης Στυλιανάκος (stilia.johny)
[stilia.johny@gmail.com]
Χρήστος Τριανταφύλλης (clepto)
[christriant1995@gmail.com]
Γιώργος Καντιάνης (GeorgeTheSuper)
[georgethesuper@hotmail.gr]
Ηλίας Σταμάτης (Ilias95)
[stamatis.iliass@gmail.com]

Σελιδοποίηση - Γραφικά

Μάριος Παπαχρήστου (MaR1oC)
[mrmarios97@gmail.com]
Γιάννης Πανταζόπουλος (HardCotton)
[admin@hard-cotton.gr]

Επιμέλεια Κειμένων

Νίκος Αλμπανόπουλος (nikosal)
[nikosal@freemail.gr]
Μάριος Τσιάντας (dtrzG)
[marios.tsiantas@gmail.com]
Μάρα Σδράκα (parenthesis)
[paren8esis@gmail.com]

Ubuntistas

#15 editorial

UBUNTISTAS 15 τευχών – Τι νομίζετε ;
Προχωράμε και μεγαλώνουμε.

Το καλοκαίρι ήρθε και πέρασε, διακοπές, παραλία, θάλασσα κάπου εδώ μας άφησαν.. Κι εμείς εδώ ξεκούραστοι πλέον από την καλοκαιρινή μας ανάπαυλα, φτάσαμε αισίως στο τεύχος 15 του περιοδικού της κοινότητάς μας.
Αρκετά γεγονότα έχουν συμβεί μέχρι στιγμής, άλλα λιγότερης σημασίας και άλλα μεγαλύτερης, που νιώθω υποχρεωμένος να αναφέρω.
Ξεκινώ πρώτα πρώτα με τα του "οίκου μας" και φυσικά αναφέρομαι στις αποχωρήσεις από τις θέσεις των συντονιστών κάποιων ακούραστων εθελοντών για την ομαλή και σωστή λειτουργία της κοινότητάς μας..
Θέλω να εκφράσω ένα μεγάλο ΕΥΧΑΡΙΣΤΩ στα διαδικτυακά φιλαράκια (και για κάποιους όχι μόνο διαδικτυακά) :
Epirotes, filippos.xf, ftso, logari81, sokoban4ever για την βοήθεια που μας πρόσφεραν. Είμαστε σίγουροι οτι θα παρακολουθούν όποτε μπορούν.
Άφησα σκόπιμα σε δεύτερη μοίρα ένα τεράστιας σημασίας γεγονός (γιατί χωρίς τους ενεργούς χρήστες κοινότητα δεν θα υπήρχε) και δεν είναι άλλο από την επιτυχή επανέγκριση της ελληνικής κοινότητας από το Ubuntu LoCo Council (συμβούλιο τοπικών ομάδων Ubuntu).
Η αίτηση της κοινότητας εξετάστηκε την Τρίτη 19 Ιουνίου 2012 στις 11.00 η ώρα το βράδυ τοπική ώρα Ελλάδος.
Αντιγράφω κάποια ουσιώδη σημεία που έχει γράψει ο Geochr για την σημασία που έχει η επανέγκριση :

" Η σελίδα αυτή είναι ο καθρέπτης του έργου της κοινότητας. Με βάση αυτή τη σελίδα το συμβούλιο τοπικών ομάδων Ubuntu θα αξιολογήσει και θα αποφασίσει αν αξίζουμε να είμαστε μία από τις επίσημες ομάδες του Ubuntu... Ενδεικτικά αναφέρω ότι μόνο οι επίσημες κοινότητες έχουν κάποια προνόμια, όπως δωρεάν cd, υλικά για συνέδρια και άλλα υλικά που κατά καιρούς δίνονται στις κοινότητες. Ας βοηθήσουμε όλοι, ο καθένας με τον τρόπο του, να πετύχουμε την 3η συνεχόμενη επανέγκριση της κοινότητας !!! "

Κάτι που επίσης είναι σημαντικό, είναι να βοηθήσουμε όσο μπορούμε να γίνει και η κυπριακή κοινότητα επίσημη κοινότητα Ubuntu.
Είχαμε και στο περιοδικό μας κάποιες αλλαγές προσώπων που θα φανούν από το τεύχος αυτό, όπως τη συμμετοχή νέων συντακτών.
Μπορείτε να διαβάσετε για το LibreOffice, για το Kiku, ένα χρήσιμο πρόγραμμα αναγνώρισης φωνής, μια πρώτη γνωριμία με το FF multi converter με το οποίο μπορούμε να μετατρέψουμε εύκολα αρχεία ήχου, βίντεο, εικόνας όπως επίσης και έγγραφα κειμένου σε όλα τα δημοφιλή format, μια παρουσίαση με πολλά γκάζια και βέβαια εννοώ το Torcs, ένα εξαιρετικό racing game που φυσικά είναι ελεύθερου λογισμικού καθώς και για τη συμμετοχή της ελληνικής κοινότητας Ubuntu σε δημοτική εκδήλωση στην Ανατολική Αττική το Σάββατο 19 Μαΐου.

Ευχόμαστε σε όλους,
Καλή Ανάγνωση!
Γιώργος Μακρής - ubuderix

ΠΕΡΙΕΧΟΜΕΝΑ

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 αυτοοργάνωση 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 Γιώργου Χριστοφή

Νέα της Ελληνικής Κοινότητας Ubuntu Linux

Επανέγκριση ελληνικής κοινότητας Ubuntu

Όπως θα γνωρίζετε η ελληνική κοινότητα Ubuntu (Ubuntu Greece), είναι η επίσημη και αναγνωρισμένη κοινότητα Ubuntu για την Ελλάδα. Κάθε δύο χρόνια όλες οι αναγνωρισμένες ομάδες Ubuntu όπως η δική μας, πρέπει να παίρνουν την επανέγκριση τους από το Ubuntu LoCo Council (συμβούλιο τοπικών ομάδων Ubuntu). Έτσι λοιπόν ήρθε η στιγμή για τη δεύτερη επανέγκριση της κοινότητάς μας η οποία πραγματοποιήθηκε στις 19 Ιουνίου 2012.

Η προετοιμασία γι' αυτή τη μέρα είχε ξεκινήσει από τις αρχές Μαΐου όπου και ξεκινήσαμε να συντάσσουμε τη σχετική σελίδα [1] της αίτησής μας. Στη σελίδα αυτή καταγράψαμε όλο το έργο και τις δραστηριότητες που είχε η κοινότητά μας από το 2010 όπου και είχε πραγματοποιηθεί η προηγούμενη επανέγκρισή της έως τον Μάιο 2012. Με βάση λοιπόν αυτή τη σελίδα που δημιουργήθηκε την Τρίτη 19 Ιουνίου στις 23:00 ώρα Ελλάδος (20:00 UTC), εξετάστηκε η αίτηση επανέγκρισης της ελληνικής κοινότητας Ubuntu από το συμβούλιο τοπικών κοινοτήτων Ubuntu (LoCo council) στο κανάλι #ubuntu-meeting στο IRC.

Μια σύντομη παρουσίαση [2] της κοινότητας και της δράσης της, ήταν αρκετή για το συμβούλιο να πειστεί ότι η ελληνική κοινότητα Ubuntu είναι μία ενεργή και ζωντανή κοινότητα, αποτελούμενη από μέλη με όρεξη και κέφι για το Ubuntu. Εξαιρετικά σχόλια απέσπασε η σελίδα της αίτησής μας και αξιολογήθηκε ως μια πλήρης και καλά δομημένη σελίδα. Το έργο της κοινότητας και των μελών της κρίθηκε με αρκετά καλά σχόλια και χωρίς παρατηρήσεις από τα μέλη του LoCo council κι έτσι η κοινότητα θα είναι και για τα επόμενα 2 χρόνια η επίσημη και αναγνωρισμένη κοινότητα Ubuntu στην Ελλάδα!

Η ελληνική κοινότητα Ubuntu ξεκίνησε τη δραστηριότητά της το 2005 και από το 2008 όπου και αναγνωρίστηκε για πρώτη φορά από το LoCo council, διατηρεί τον τίτλο της επίσημης κοινότητας Ubuntu χάρις το έργο και τις δραστηριότητες των μελών της. Αξίζουν πολλά μπράβο σε όλους όσους βοηθούν και συμμετέχουν στην κοινότητα.

[1] https://wiki.ubuntu.com/GreekTeam/ReApprovalApplication2012

[2] http://irclogs.ubuntu.com/2012/06/19/%23ubuntu-meeting.html#t20:09

Το περιοδικό Ubuntistas στο κέντρο λογισμικού

Ύστερα από την ιδέα κάποιων μελών της κοινότητας αλλά και με αφορμή μια σχετική συζήτηση στο φόρουμ για την ένταξη του περιοδικού μας στο Κέντρο λογισμικού του Ubuntu, έγιναν οι απαραίτητες ενέργειες ώστε το περιοδικό να είναι διαθέσιμο στο Κέντρο λογισμικού. Εδώ και λίγους μήνες το περιοδικό Ubuntistas της ελληνικής κοινότητας Ubuntu είναι διαθέσιμο για τους αναγνώστες του, εκτός από την ιστοσελίδα του [1] και μέσα από το κέντρο λογισμικού του Ubuntu.

Αυτή τη στιγμή στο κέντρο λογισμικού βρίσκονται οι δυο τελευταίες κυκλοφορίες του περιοδικού, τα τεύχη 13 και 14, ενώ από εδώ και στο εξής όλα τα επόμενα τεύχη θα είναι διαθέσιμα και στο κέντρο λογισμικού!

Τα προηγούμενα τεύχη μπορείτε να τα βρείτε μόνο στην ιστοσελίδα του περιοδικού, ενώ κάθε νέα κυκλοφορία τεύχους θα είναι διαθέσιμη και με τους δύο τρόπους.

Το περιοδικό Ubuntistas μπορείτε να το βρείτε εύκολα κάνοντας μια απλή αναζήτηση στο κέντρο λογισμικού των εκδόσεων 11.04, 11.10 και 12.04.

[1] http://ubuntistas.ubuntu-gr.org

του Γιώργου Μακρή

Παρουσίαση ΕΛ/ΛΑΚ - Ubuntu στο Κορωπί

Ο Δήμος Κορωπίου διοργάνωσε μια εκδήλωση με ελεύθερη συμμετοχή όσων μπορούσαν να παρουσιάσουν κάτι σχετικό με ΕΛ/ΛΑΚ και Linux.

Αφού ήρθαμε σε επικοινωνία με τους διοργανωτές είπαμε οτι ευκαιρία είναι επιτέλους να ακουστούν παραέξω αυτά που είναι γνωστά σε εμάς , σε άλλους που δεν έχουν ακούσει τίποτε.

Με πολύ γρήγορη συνεννόηση και αφού είχε αναρτηθεί στο φόρουμ μας σχετικό νήμα για την εκδήλωση – πρόσκληση, δόθηκε ραντεβού για τις 19 Μαΐου στο Κορωπί.

Σε μια αίθουσα του Πνευματικού Κέντρου και παρουσία κάποιων δημοτών της πόλης του Κορωπίου, ξεκίνησε η εκδήλωση.

Έγινε μια εισαγωγή από τους ανθρώπους που είχαν την ιδέα για το όλο εγχείρημα που θέλουν να φέρουν εις πέρας στην πόλη του Κορωπίου.

Εικόνα 1 - Διοργανωτές εκδήλωσης

Οι οικοδεσπότες ήταν πολύ φιλικοί και με αρκετή διάθεση να ακούσουν και να μάθουν για το ΕΛ/ΛΑΚ, για το Linux και φυσικά για το τι εστί Ubuntu.

Την παρουσίαση από μεριάς μας δεν θα μπορούσε να την κάνει άλλος από τον κατά κόσμο Σίμο Ξενιτέλη.

Εικόνα 2 - Παρουσίαση UBUNTU από simosx

Ο Σίμος με το υλικό που είχε μαζί του εξήγησε με απλά λόγια τι είναι το ΕΛ/ΛΑΚ , τι μας προσφέρει, τι είναι το Linux και φυσικά μίλησε για τη διανομή Ubuntu και έδειξε πως υπάρχει και κάτι άλλο πέρα από την Microsoft.

Από το φόρουμ της ελληνικής κοινότητας Ubuntu παρευρέθησαν ο simosx , ο Geochr και ο ubuderix.

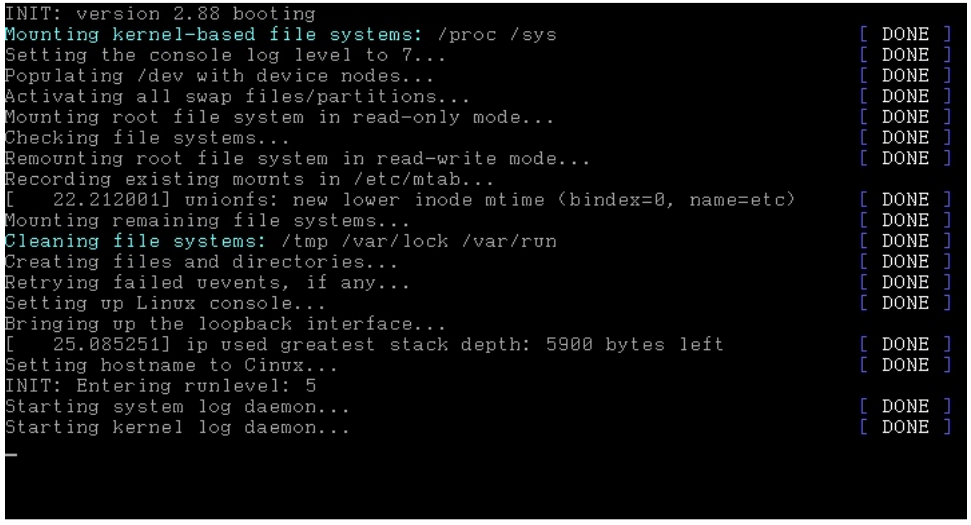
Εικόνα 3 - UBUNTEROS & ΔΙΟΡΓΑΝΩΤΕΣ

Αυτά προς το παρόν, το σίγουρο είναι οτι θα υπάρξει και συνέχεια.

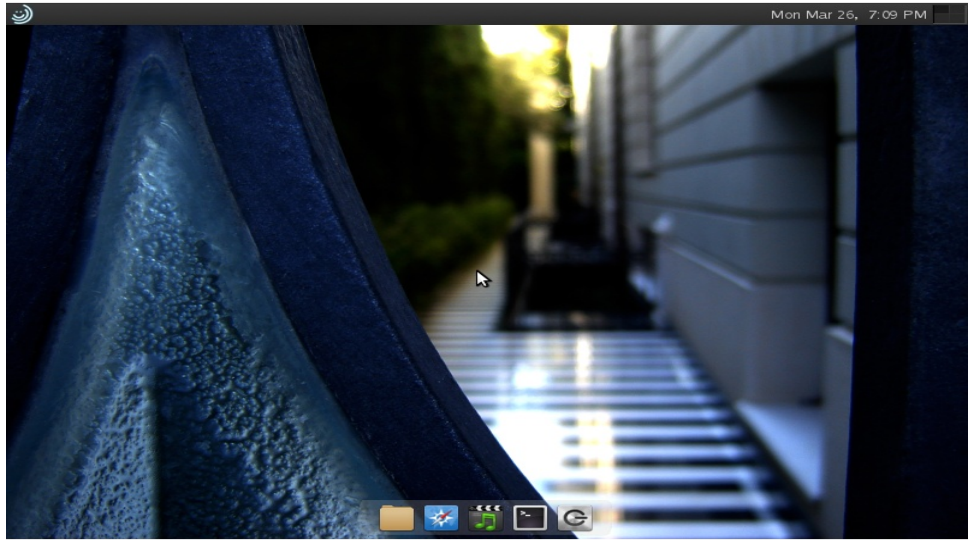
του Κωνσταντίνου Αποστόλου

Cinux

Cinux. Τι είναι το Cinux; Πολλοί θα έχετε ακούσει (ή όχι) αυτό το όνομα αλλά δεν θα ξέρατε τι είναι. Το Cinux λοιπόν, είναι ένα λειτουργικό σύστημα βασισμένο στον πυρήνα του Linux. Είναι 100% ελεύθερο, 100% ελληνικό και 100% "ανεξάρτητο" λογισμικό. Με τον όρο ανεξάρτητο αναφέρομαι στο γεγονός ότι δεν είναι βασισμένο σε κάποια γνωστή διανομή, αλλά στον εαυτό της. Η ανάπτυξή του ξεκίνησε μόλις το Σεπτέμβριο του 2010 και η πρώτη του έκδοση δημοσιοποιήθηκε τον Απρίλιο του 2011. Για την ανάπτυξή της χρησιμοποιήθηκε το LFS (Linux From Scratch) και BLFS (Beyond LFS) και τώρα προς το τέλος και το CLFS (Cross LFS), σαν έρευνα για υποστήριξη περισσότερων πλατφορμών, όπως εκείνη του Raspberry Pi.



Η πρώτη κυκλοφορία του Cinux (0.4) δεν είχε καν γραφικό περιβάλλον, μα μόνο το τερματικό του. Δεν άργησε όμως και το γραφικό περιβάλλον να κάνει την εμφάνισή του και μια εβδομάδα μετά κυκλοφόρησε το Cinux 1.0 με GNOME 2.30.2. Από εκεί και πέρα προστέθηκαν διάφορες εφαρμογές που σκοπό είχαν να κάνουν τη ζωή του χρήστη ευκολότερη.

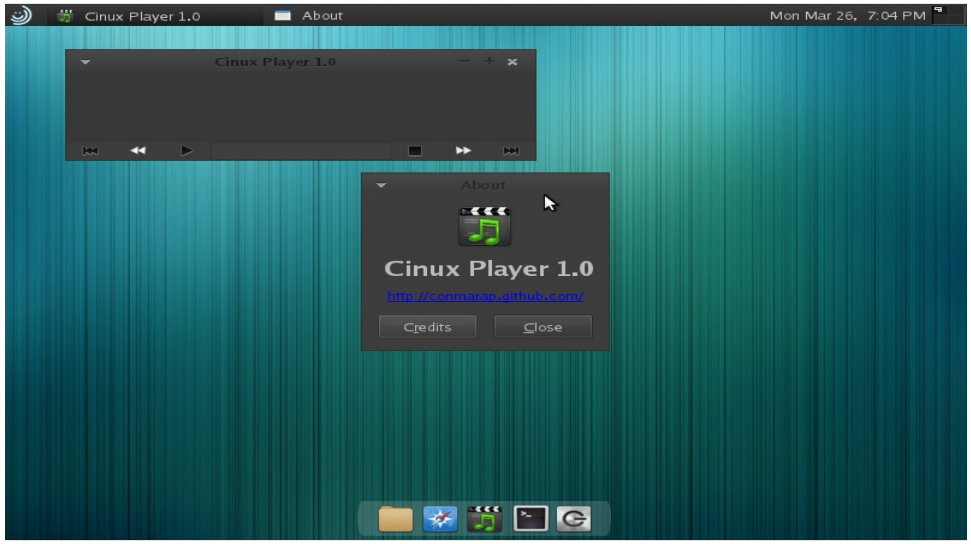


Η πιο πρόσφατη έκδοση του Cinux είναι η 2.0 (Pandora) η οποία κυκλοφόρησε το Μάρτιο του 2012, ένα μήνα πριν κλείσει τον πρώτο χρόνο διαθεσιμότητας στο κοινό. Για να φτάσει σε αυτό το σημείο, η διανομή πέρασε διάφορες και μεγάλες αλλαγές και αρχικώς ήταν προγραμματισμένο να κυκλοφορήσει στα μέσα Φεβρουαρίου.

Μία από τις αισθητές αλλαγές, είναι το γεγονός οτι πλέον πολλές καθημερινά χρησιμοποιούμενες εφαρμογές υποστηρίζονται "out of the box" από το Cinux 2.0 : Google Chrome, Dropbox, Skype, κτλ. Από εκεί και πέρα, υπάρχει το Cinux Get, ο δικός του διαχειριστής πακέτων, αλλά και το Dpkg για υποστήριξη πακέτων από το Ubuntu και Debian. Επίσης, δεν υπάρχει προεπιλεγμένη γλώσσα προγραμματισμού, αυτή που να αντιπροσωπεύει τη διανομή δηλαδή, αλλά επιλογές όπως οι Vala, Python, C#, C++, C,

Perl, Ruby, και όσες ακόμα βάζει ο νους σας, κι αυτό για να δοθεί η δυνατότητα στο χρήστη να "παίξει" και να χαρεί.

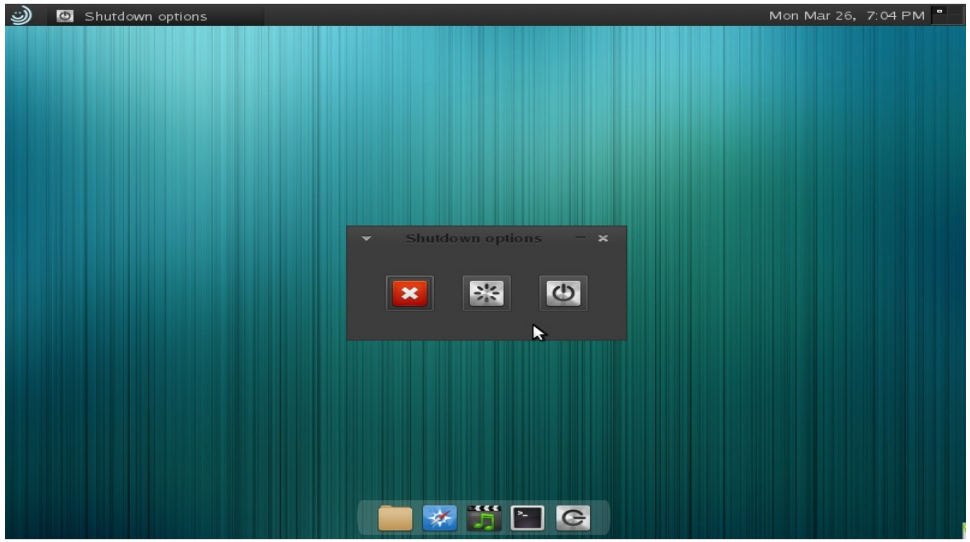
Για να φανεί η χρησιμότητα των γλωσσών προγραμματισμού που έρχονται μαζί με το Cinux 2.0 έχουν δημιουργηθεί διάφορες εφαρμογές και έχουν πακεταριστεί στο ISO του Pandora. Η πρώτη από αυτές είναι ο Session Manager, η εφαρμογή που δίνει τη δυνατότητα στο χρήστη να κάνει επανεκκίνηση ή τερματισμό τον υπολογιστή του. Μία ακόμη εφαρμογή είναι ο Cinux Player, ο music player του Cinux 2.0. Πέρα από αυτές, έχει γραφεί και η εφαρμογή About Cinux σε Python αλλά και ένα μεγαλύτερο πρόγραμμα, το JetBrowser, ένας webkit περιηγητής διαδικτύου που σκοπό έχει να αντικαταστήσει τον Firefox, τον προεπιλεγμένο περιηγητή στο Cinux.

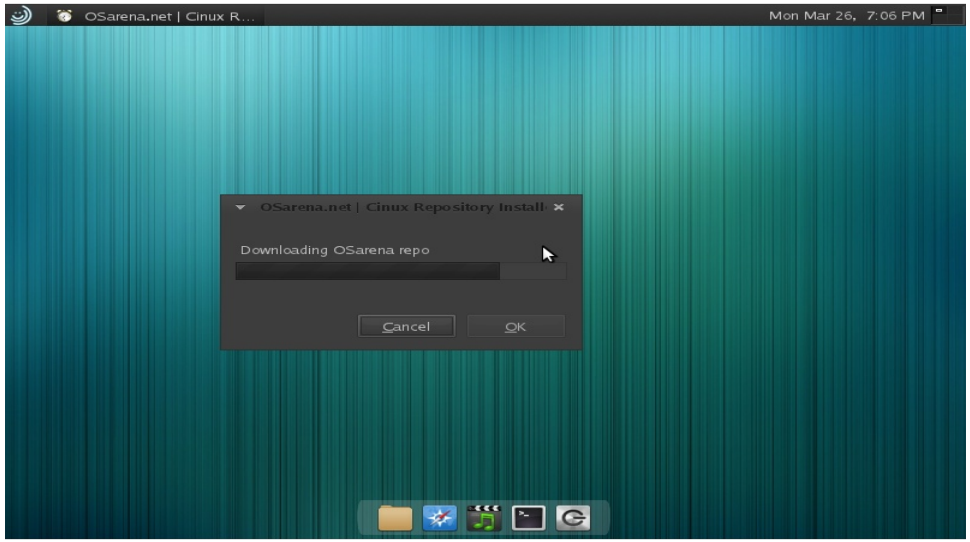


Ο σκοπός του Cinux είναι να προσφέρει μια πλήρη, γρήγορη και ασφαλή πλατφόρμα στο χρήστη, ότι κι αν είναι εκείνος : φοιτητής, εκπαιδευτικός, προγραμματιστής, IT Manager κτλ. Αλλά γενικά προορίζεται για τον εκπαιδευτικό τομέα.

Γιατί ο υπολογιστής είναι εύκολος και ο καθένας έχει το δικαίωμα να τον "μάθει".

Τη διανομή μπορείτε να την κατεβάσετε από το http://mycinux.blogspot.com/ από όπου και μπορείτε να διαβάζετε νέα γύρω από αυτό.



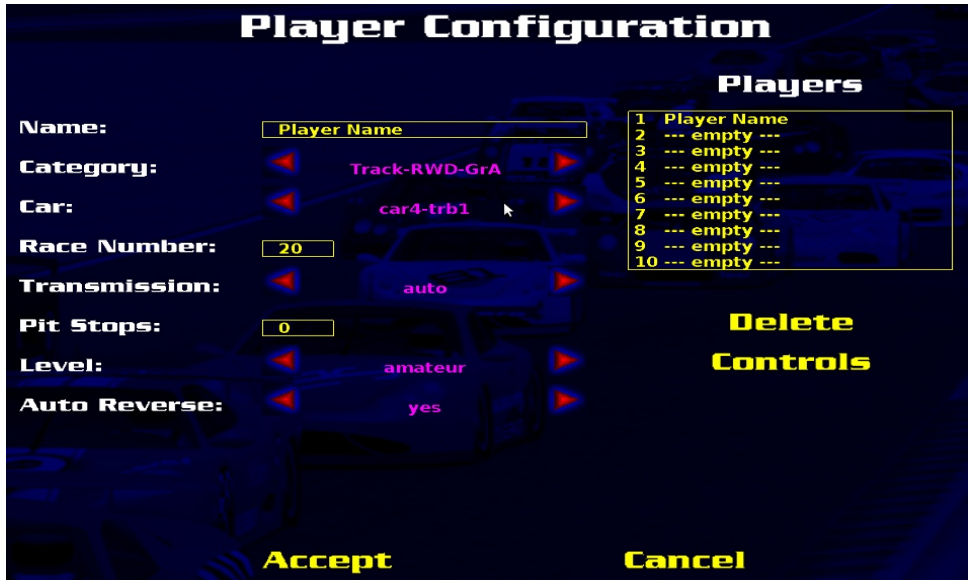


Torcs

Ένα από τα καλύτερα παιχνίδια ελεύθερου λογισμικού. Το Torcs, παιχνίδι αγώνων, δε θα λέγαμε ότι θα μπορούσε να ανταγωνιστεί τα αγωνιστικά ηλεκτρονικά παιχνίδια αλλά έχει αρκετά καλά γραφικά για τα δεδομένα των περισσότερων παιχνιδιών του Ubuntu. Όμως, ας το δούμε βήμα-βήμα αναλυτικά:

Ξεκινώντας, επιλέγουμε Race για να ξεκινήσουμε κάποιο πρωτάθλημα ή γρήγορο αγώνα. Επιλέγουμε Configure Players για να επεξεργαστούμε διάφορες επιλογές, όπως βλέπουμε στην παρακάτω εικόνα.

Μπορούμε να αλλάξουμε το όνομά μας, το αυτοκίνητο, τον αριθμό αγώνων (σε περίπτωση που ξεκινήσουμε πρωτάθλημα), επίπεδο κτλ. Στο Options, μπορούμε να αλλάξουμε προεπιλεγμένες ρυθμίσεις γραφικών, ήχου κτλ.



Τώρα, ας δούμε και το gameplay. Υπάρχει μεγάλη ποικιλία από οχήματα, είτε σχεδιασμένα από τους developers, είτε εμπνευσμένα από μάρκες αυτοκινήτων όπως BMW και Peugeot.

Όπως διαπιστώνετε έχει αρκετά καλά γραφικα και gameplay, διότι υπάρχουν επιλογές για πρωτάθλημα, γρήγορο αγώνα και προσαρμοσμένοι αγώνες σε πίστες που έχετε σχεδιάσει οι ίδιοι! Επίσης κάθε χρόνο πραγματοποιείται το TORCS Endurance World Championship 2012. Ήδη έχει πραγματοποιηθεί ο πρώτος αγώνας στις 16 Μαΐου. Για πληροφορίες σχετικά με το πρωτάθλημα αλλά και γενικότερα, το συγκεκριμένο παιχνίδι, επισκεφτείτε το site: http://torcs.sourceforge.net/.

Είναι πραγματικά ενα πολύ ενδιαφέρον, αλλά και αρκετά δύσκολο (ειδικά στο χειρισμό του αυτοκινήτου στις στροφές) παιχνίδι! Έχει updates αρκετά συχνά και συνεχώς αποκτά νέες πίστες και αυτοκίνητα.

Μπορείτε να κάνετε λήψη την έκδοση 1.3.3 από το Ubuntu Software Center αλλά και από την ιστοσελίδα: http://torcs.sourceforge.net/

Ελπίζω οι οδηγίες να σας φάνηκαν χρήσιμες....

του Νίκου Βασιλειάδη

Open Source Ecology

Τι είναι;

Το Open Source Ecology είναι ένα δίκτυο από αγρότες, μηχανικούς και υποστηρικτές οι οποίοι τα τελευταία δύο χρόνια έχουν αφοσιωθεί στη δημιουργία του Global Village Construction Set. Το οποίο στην ουσία είναι μια ανοιχτού κώδικα, χαμηλού κόστους και υψηλής απόδοσης τεχνολογική πλατφόρμα που σκοπό έχει να διευκολύνει την κατασκευή των 50 πιο χρήσιμων βιομηχανικών μηχανών που μπορούν να μας επιτρέψουν τη δημιουργία ενός βιώσιμου πολιτισμού με σύγχρονες ανέσεις. Το GVCS χαμηλώνει τον πήχη για όσους θέλουν να ξεκινήσουν να ασχολούνται με την αγροτική παραγωγή, τον κατασκευαστικό τομέα, τη βιομηχανία και μπορεί να γίνει αντιληπτό ως ένα σύνολο αρθρωτών εργαλείων τόσο ευέλικτο όσο τα lego που μπορεί να δημιουργήσει ολόκληρες οικονομίες, είτε στο Missouri, όπου το έργο αυτό ξεκίνησε, είτε στην αστική ανάπλαση, ή ακόμα και στον αναπτυσσόμενο κόσμο.

Εικόνα 1 - GCVS

Ποιες είναι οι αρχές σχεδιασμού των εργαλείων του GVCS;

Τα διάφορα εργαλεία που απαρτίζουν το GVCS σχεδιάζονται με τις ακόλουθες προϋποθέσεις:

1. Ανοιχτός Κώδικας: Τα εργαλεία που αναπτύσσονται και οι τρισδιάστατες απεικονίσεις τους είναι ελεύθερα προσβάσιμα στις ιστοσελίδες του OSE με οδηγίες κατασκευής και χρήσης ενώ και τα οικονομικά στοιχεία του έργου μαζί με λίστες υλικών δημοσιεύονται αναλυτικά στο διαδίκτυο.

2. Χαμηλό κόστος: Τα εργαλεία που μέχρι στιγμής έχουν αναπτύξει έχουν κατά μέσο όρο 8 φορές λιγότερο κόστος από αντίστοιχα εμπορικά εργαλεία συνυπολογίζοντας εργατικό κόστος 15 δολαρίων ανά εργατοώρα.

3. Τμηματικός σχεδιασμός: Μηχανές, στοιχεία των εργαλείων, κτλ είναι σχεδιασμένα ώστε να μπορούν να αλλάξουν εύκολα, κάποια να λειτουργήσουν μόνα τους ή σε συνδυασμό με άλλα.

4. Επισκευαζόμενα από το χρήστη: Όλα τα εργαλεία είναι σχεδιασμένα με τέτοιο τρόπο ώστε ο χρήστης να μπορεί να τα αποσυναρμολογήσει για να τα συντηρήσει, να τα επισκευάσει και να τα συντηρήσει χωρίς την ανάγκη χρήσης εξουσιοδοτημένου τεχνικού.

5. Κάντε το μόνοι σας: Ο χρήστης μπορεί να φτιάξει τα εργαλεία μόνος του ακολουθώντας τις οδηγίες ή ακόμη και να παρέμβει στα σχέδια ώστε να καλύπτουν τις ανάγκες του.

6. Κατασκευή Κλειστού Βρόγχου: Ο στόχος (όταν ολοκληρωθεί) είναι, χρησιμοποιώντας κανείς τα παρεχόμενα από το GVCS εργαλεία, να είναι σε θέση να φτιάξει το δικό του σύνολο εργαλείων χωρίς να χρειάζεται εξωτερικούς πόρους (φανταστείτε το σαν ένα self replicating μηχάνημα όπως το RepRap).

7. Υψηλή απόδοση: Η απόδοση των εργαλείων θα πρέπει να είναι ανάλογη των εμπορικών προκειμένου το GVCS να είναι βιώσιμο, και για ορισμένα από αυτά τα εργαλεία οι επιδόσεις τους ξεπερνάνε κατά πολύ εμπορικές υλοποιήσεις.

8. Ευέλικτη κατασκευή: Η ευέλικτη κατασκευή με εργαλεία γενικής χρήσης μπορεί σε μικρές κλίμακες παραγωγής να είναι μια βιώσιμη εναλλακτική συγκριτικά με την συγκεντρωτική παραγωγή μεγάλης κλίμακας.

9. Διανεμητικά Οικονομικά: Η δημιουργία επιχειρήσεων γύρω από το GVCS ενθαρρύνεται, και θεωρείται φυσική.

10. Βιομηχανική απόδοση: Προκειμένου το GVCS γίνει βιώσιμο, θα πρέπει η απόδοση της πλατφόρμας GVCS να είναι ανάλογη των αντίστοιχων βιομηχανικών .

Τα περισσότερα από τα συστήματα που αναπτύσσονται στα πλαίσια του GVCS έχουν ανταλλακτικά που βασίζονται σε εμπορικές κατασκευές. Ο στόχος είναι σταδιακά όλο και περισσότερα τμήματα των κατασκευών αυτών να αντικαθίστανται με ανοιχτού κώδικα λύσεις και σχέδια.

Τα μηχανήματα:

Power Cube: Είναι μια μονάδα παροχής ισχύος που υλοποιείται από μια μηχανή εσωτερικής καύσης και μια υδραυλική αντλία (για την παροχή ενέργειας και σε άλλες συσκευές) και χρησιμοποιείται από το LifeTrac και το MicroTrac ως μονάδα παροχής ισχύος. Στην παρούσα φάση, όπως πολλά από τα σχέδια του GVCS, το Power Cube απαρτίζεται από εμπορικά προϊόντα, όπως η μηχανή εσωτερικής καύσης και η υδραυλική αντλία και ουσιαστικά μόνο το πλαίσιο είναι σχέδιο του GVCS.

Εικόνα 2

Drill Press: Είναι μια πρέσα για τη δημιουργία τρυπών διαμέτρου μιας ίντσας τουλάχιστον σε φύλλα μετάλλου χωρίς να χρειάζεται χρήση τρυπανιού. Χρησιμοποιείται για να αυξηθεί η ταχύτητα παραγωγής.

Compressed Earth Brick Press (γνωστό και ως The Liberator): Είναι από τα πλέον ολοκληρωμένα συστήματα που έχουν αναπτυχθεί στα πλαίσια του GVCS, ουσιαστικά είναι μια πρέσα παραγωγής τούβλων από χώμα. Το εν λόγω σύστημα έχει τη θεωρητική ικανότητα παραγωγής 16 τούβλων το λεπτό (αν καταφέρετε να το τροφοδοτείτε με χώμα τόσο γρήγορα), παράγοντας τούβλα με αντοχή 700-1000 psi. Η απόδοσή του είναι πολύ μεγαλύτερη από τα περισσότερα αντίστοιχα εμπορικά μηχανήματα που πωλούνται σε πολλαπλάσιες τιμές. Στο μέλλον, στο Liberator θα ενσωματωθεί ένα σύστημα ελέγχου που θα έχει βάση το Arduino με στόχο την μεγαλύτερη αυτοματοποίηση της λειτουργίας του. Παράλληλα, θα αρχίσει και η κατασκευή του ως

προϊόν που θα μπορεί να αγοράσει κανείς για τη χρηματοδότηση του έργου.

Εικόνα 3 - CEB Press

Σε πολλά μέρη του κόσμου τα τούβλα από τοπικό χώμα χρησιμοποιούνται για την κατασκευή κτιρίων, δαπέδων, ακόμη και πεζοδρόμων ενώ και στην Ελλάδα χρησιμοποιούνται (από όσο έχω δει με χειροκίνητες πρέσες) σε παραδοσιακά κτίσματα στην ηπειρωτική και νησιωτική χώρα.

LifeTrac: Και φυσικά τι θα ήταν μια φάρμα χωρίς το δικό της τρακτέρ. Σίγουρα δεν πρόκειται για τα πλέον άνετα και διαστημικά τρακτέρ που μπορεί κανείς να συναντήσει σε μια σύγχρονη αγροτική μονάδα. Ο σπαρτιατικός σχεδιασμός του έχει σαν στόχο τη δημιουργία ενός οχήματος υψηλής απόδοσης, που θα αντέχει στο χρόνο, όπως και το Liberator. Το LifeTrac θα αρχίσει σύντομα να παράγεται ώστε να χρηματοδοτήσει το έργο.

Σίγουρα, με μια πρώτη ματιά δεν παρέχει τις ανέσεις που παρέχει ένα σύγχρονο τρακτέρ, ούτε καν από τα στοιχεία της φύσης. Όμως, από την άλλη, το συγκριτικά χαμηλό κόστος του και η δυνατότητα παραγωγής και συντήρησης σε τοπικό επίπεδο ίσως το κάνει μια βιώσιμη λύση για κάποιους αγρότες.

Εικόνα 4 - LifeTac III

Εικόνα 5 - LifeTac with tracks

MicroTrac: Είναι το μικρό αδελφάκι του LifeTrac, ένα μικρό τρακτέρ που ταιριάζει σε μικρότερες αγροτικές μονάδες αλλά και εργασίες που το LifeTrac δεν είναι απαραίτητο. Ουσιαστικά πρόκειται για μια μικρή έκδοση του LifeTrac με τη δυνατότητα να παίρνει μια σειρά από εργαλεία που χρησιμοποιεί και το LifeTrac.

Soil Pulverizer: Είναι ένα πρόσθετο για το LifeTrac που έχει σαν στόχο να μετατρέπει το έδαφος σε χώμα ώστε να χρησιμοποιηθεί από το Liberator.

Στα άμεσα σχέδιά τους είναι και η δημιουργία του ακόλουθου:

Εικόνα 6 - CNC Torch Table

CNC Torch Table : Πρόκειται για ένα ειδικό τραπέζι κοπής μετάλλου σε τρεις διαστάσεις (XYZ), το οποίο είναι πλήρως αυτοματοποιημένο και ελεγχόμενο από υπολογιστή (ο οποίος τρέχει Linux φυσικά). Το πρωτότυπο που είχαν φτιάξει δυστυχώς δεν λειτούργησε όπως έπρεπε καθώς υπήρχε παρεμβολή στα ηλεκτρονικά του από τον κόφτη πλάσματος. Όταν ολοκληρωθεί, θα έχει επιλυθεί το πρόβλημα και θα χρησιμοποιηθούν ανοιχτού κώδικα ελεγκτές των κινητήρων βήματος.

Όταν ολοκληρωθεί θα δώσει την δυνατότητα στην ομάδα να παράγει τα δικά της μεταλλικά αντικείμενα χωρίς να καταφεύγει σε κάποιον μηχανουργό.

Πώς μπορεί κανείς να συμμετάσχει;

Το πρώτο και πιο απλό πράγμα που μπορεί να κάνει κανείς και χωρίς να έχει εξειδικευμένες γνώσεις είναι να βοηθήσει στη συμπλήρωση και συντήρηση του wiki το οποίο έχει τον ρόλο μιας συνεργατικής βάσης δεδομένων και είναι το κύριο μέσο αναμετάδοσης της γνώσης. Όσοι έχουν εξειδικευμένες γνώσεις μπορούν να συνεισφέρουν θεωρητικά όσο και πρακτικά σε κάποιους από τους παρακάτω ρόλους: δημιουργοί πρωτοτύπων, συνεισφορά τεχνικής γνώσης σε εξειδικευμένα θέματα, διαχειριστές έργων, σύμβουλοι, προγραμματιστές και υποστήριξη υπολογιστικών συστημάτων, παραγωγοί ντοκιμαντέρ, καλλιτέχνες, γραφίστες, κ.α. Φυσικά για όσους δεν έχουν χρόνο για εθελοντική εργασία υπάρχει πάντα η επιλογή της οικονομικής ενίσχυσης του έργου.

Τι γίνεται στην Ευρώπη;

Στην Ευρώπη υπάρχουν τοπικές ομάδες εργασίας στην Αγγλία, Γερμανία, Ισπανία, Ιταλία, Σλοβενία, Πολωνία, Σερβία και Ελλάδα μέχρι σήμερα. Το Ευρωπαϊκό δίκτυο ασχολείται κυρίως με την ενημέρωση για την εξέλιξη των κατά τόπους δραστηριοτήτων και την προσαρμογή κάποιων πληροφοριών που διαφέρουν από την αρχική αμερικάνικη έκδοσή τους στα ευρωπαϊκά δεδομένα. Υπάρχει αντίστοιχη ιστοσελίδα καθώς και

forum συζητήσεων που βοηθούν την περαιτέρω επικοινωνία και συνεργασία μεταξύ χωρών.

Τι γίνεται στην Ελλάδα;

Στην Ελλάδα η επίσημη έναρξη των δραστηριοτήτων του δικτύου Open Source Ecology Greece έγινε τον Ιανουάριο του 2012 μετά από προσχώρηση στο OSE της ομάδας O.S.A.E.C. (Open Source Autonomous & Ecological Culture). Στόχοι του τοπικού δικτύου είναι η διεξαγωγή καμπάνιας ενημέρωσης με έντυπο και ψηφιακό υλικό, συμμετοχή σε συνέδρια και εκδηλώσεις για την αμεσότερη παρουσίαση του έργου καθώς και μετάφραση υλικού για να γνωρίσει το ελληνικό κοινό το OSE. Τα ενεργά μέλη του ελληνικού δικτύου συνεισφέρουν στην ανάπτυξη και συντήρηση του wiki και εργάζονται κυρίως σε θέματα που αφορούν τις εφαρμογές του ελεύθερου λογισμικού και το λογισμικού ανοιχτού κώδικα στο OSE. Βέβαια υπάρχουν ανεξάρτητες ομάδες που εκδήλωσαν ενδιαφέρον για κατασκευή μηχανημάτων και κυρίως του CEB Press. Μερικές από αυτές είναι το Σπιθάρι-Marathon Project (http://tinyurl.com/d9cpgb4), το Telaithrion Project (http://telaithrion.freeandreal.org/) της ομάδας Free and Real (http://www.freeandreal.org/), επίσης εκκολαπτόμενες ομάδες βρίσκονται στην Αθήνα, Εύβοια, Κρήτη, Κιλκίς, Σέρρες, Ξάνθη και την Κομοτηνή.

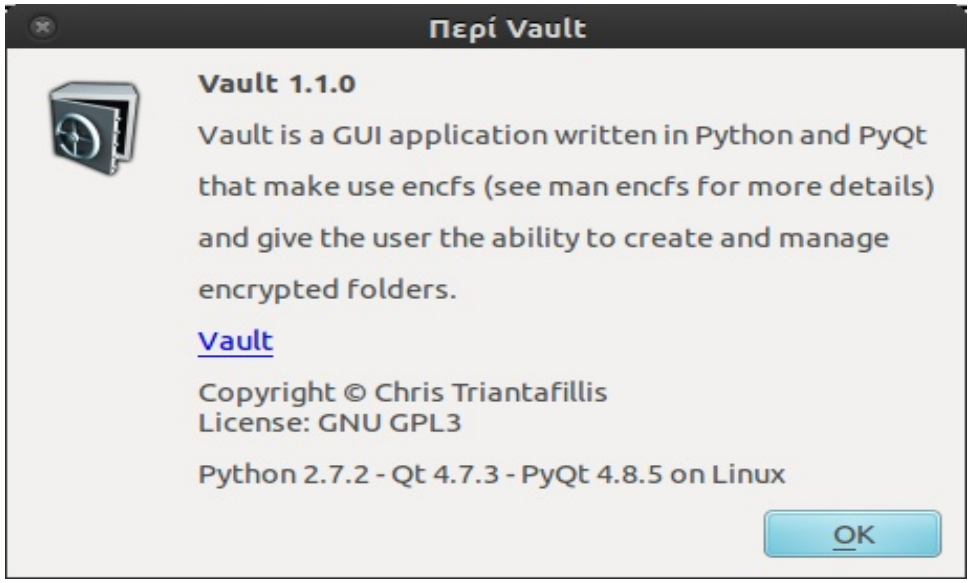
Πηγές

http://www.ted.com/talks/lang/el/marcin_jakubowski.html
http://opensourceecology.org/
http://oseeurope.org/
http://osegreece.wordpress.com/

του Χρήστου Τριανταφύλλη

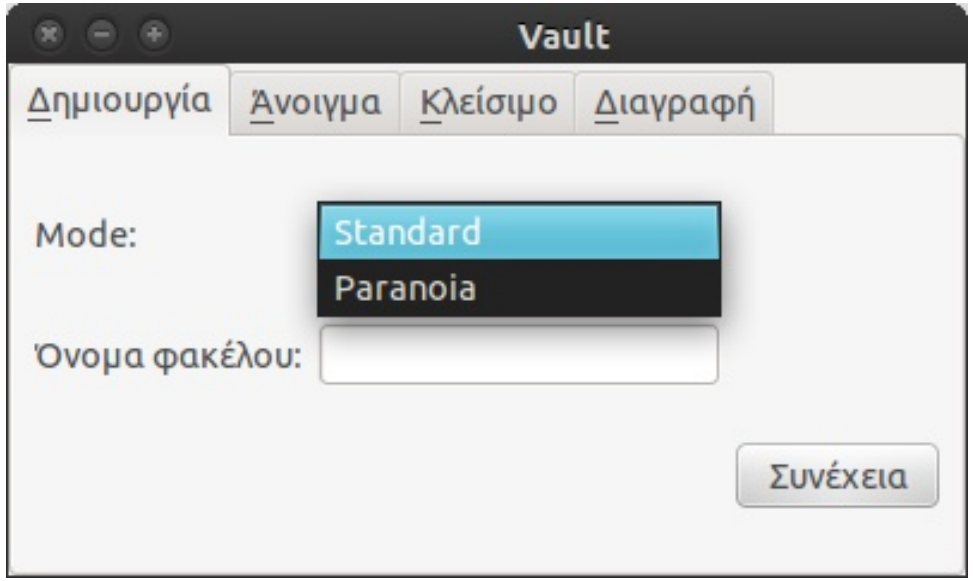
Vault

Υπήρξε ποτέ κάποια φορά που θέλατε να κρύψετε κάποια αρχεία στον υπολογιστή σας από τα αδιάκριτα μάτια; Στην περίπτωση που θελήσετε, το Vault έρχεται να σας λύσει τα χέρια! Το Vault είναι μια εφαρμογή, εμπνευσμένη από το Cryptkeeper, γραμμένη σε Python και PyQt η οποία δίνει την δυνατότητα στον χρήστη να δημιουργήσει και να διαχειριστεί κρυπτογραφημένους φακέλους.



Εικόνα vault-about.png

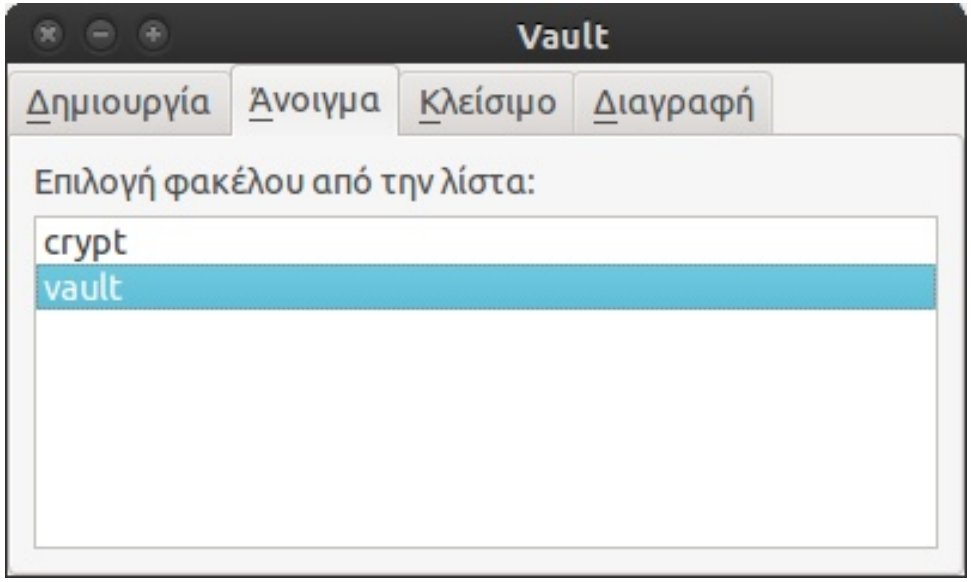
Πώς δουλεύει όμως το Vault; Χρησιμοποιεί το εργαλείο encfs (για περισσότερα, man encfs) για να κάνει την κρυπτογράφηση. Στην καρτέλα δημιουργία της εφαρμογής θα δείτε ότι υπάρχουν δύο λειτουργίες, η Standard και η Paranoia. Από το όνομα και μόνο της δεύτερης καταλαβαίνετε ότι είναι πιο ασφαλής, αν και κατά την γνώμη μου δεν χρειάζεται, μιας και η Standard είναι αρκετή.



Εικόνα vault-create.png

Αφού συμπληρώσουμε το όνομα και πατήσουμε “Συνέχεια” θα μας ζητηθεί να πληκτρολογήσουμε τον κωδικό μας και μόλις το συμπληρώσουμε και αυτό θα ανοίξει ο διαχειριστής αρχείων και έχουμε τελειώσει με την δημιουργία! Ότι αρχείο βάζετε μέσα στον φάκελο που είναι ο διαχειριστής αρχείων θα υπάρχει ταυτόχρονα και στον αντίστοιχο κρυφό (δηλαδή με το ίδιο όνομα αλλά μια τελεία μπροστά π.χ. Vault --> .Vault , για να δείτε τους κρυφούς φακέλους πατήστε Ctrl + H) αλλά θα είναι κρυπτογραφημένο. Ρίξτε μια ματιά ;)

Πιστεύω ότι είναι εύκολο να καταλάβετε τι κάνουν οι άλλες καρτέλες (όχι ότι με την παραπάνω δεν μπορούσατε).

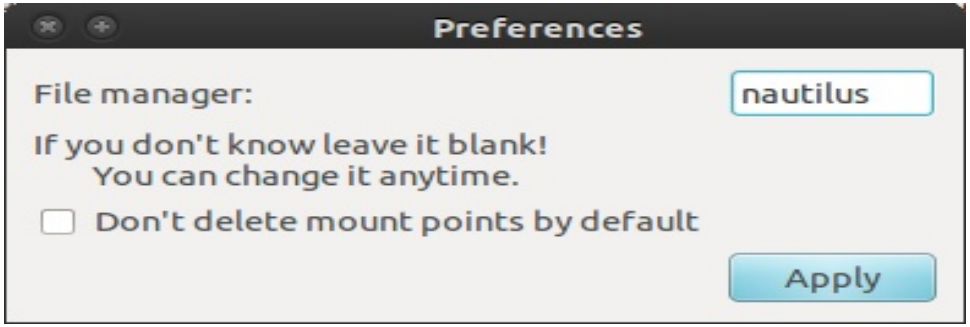


Εικόνα vault-all.png

Επίσης, έχετε την δυνατότητα να διαγράφετε τα σημεία προσάρτησης (δηλαδή τους ορατούς φακέλους στον προσωπικό σας φάκελο) μετά από κάθε κλείσιμο ή διαγραφή.

Ακόμα, αν πατήσετε δεξί κλικ στο όνομα κάποιου φακέλου στις καρτέλες “Άνοιγμα” και “Διαγραφή” θα δείτε κάποιες επιλογές όπως “Πληροφορίες” και “Αλλαγή κωδικού”.

Υπάρχει και το παράθυρο των προτιμήσεων που μπορείτε να επιλέξετε με ποιόν διαχειριστή αρχείων θα ανοίγει τους φακέλους το πρόγραμμα αλλά και να μην διαγράφει τα σημεία προσάρτησης από προεπιλογή.



Εικόνα vault-preferences.png

Για να εγκαταστήσετε το Vault μπορείτε να επισκεφθείτε την σελίδα του: http://clepto.github.com/Vault/

Το Vault είναι η πρώτη μου “κανονική” εφαρμογή και είχα σταματήσει την ανάπτυξη του εδώ και κάμποσους μήνες αλλά αποφάσισα να το ξαναγράψω σε Python3 και PyGtk!

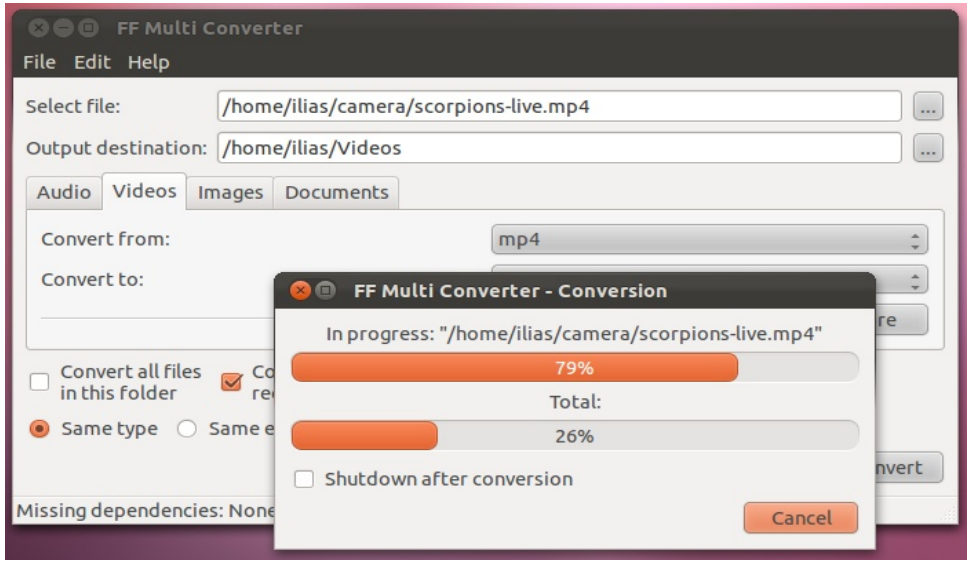
Αν κάποιος θέλει να συνεισφέρει σε επίπεδο κώδικα αλλά και σε μεταφράσεις, μπορεί να επικοινωνήσει μαζί μου στο christriant1995@gmail.com.

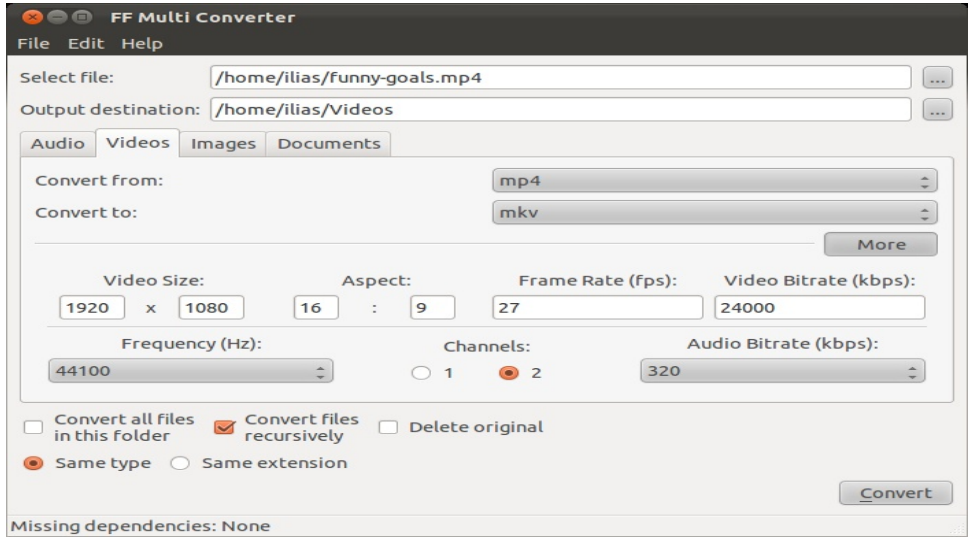
Θέλω να ευχαριστήσω τους Σάββα Radevic και Γιάννη Φυσάκη για την βοήθεια και προσφορά τους αλλά και τα μέλη της ομάδας Ubuntistas που μου έδωσαν την ευκαιρία να γράψω για την εφαρμογή μου.

του Ηλία Σταμάτη

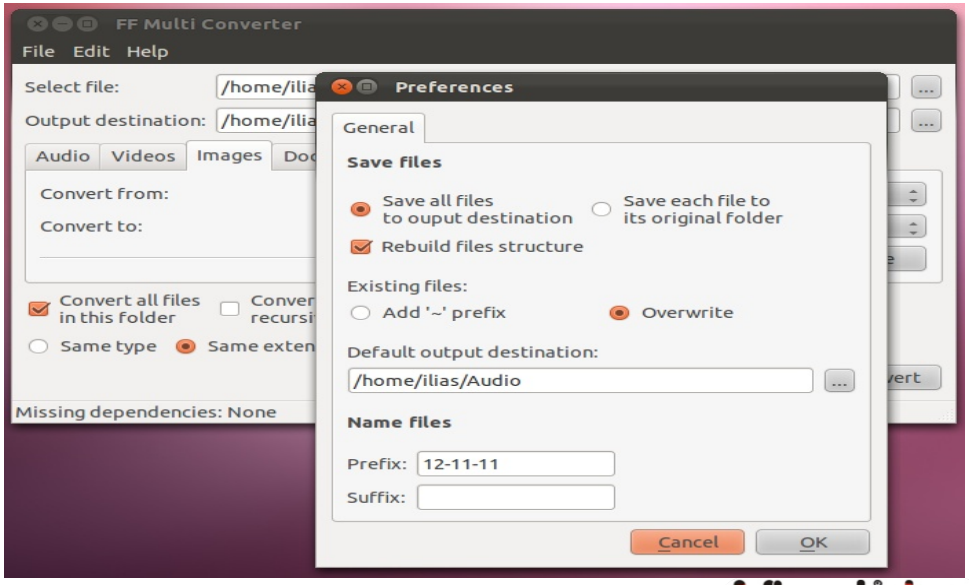
FF Multi Converter

Ο FF Multi Converter είναι μια εφαρμογή μέσω της οποίας μπορούμε να μετατρέψουμε εύκολα αρχεία ήχου, βίντεο, εικόνας, όπως επίσης και έγγραφα κειμένου σε όλα τα δημοφιλή format.





Κάποια χαρακτηριστικά της εφαρμογής είναι το πολύ εύκολο στη χρήση interface, η πρόσβαση σε κοινές επιλογές μετατροπής, η διαχείριση video ffmpeg-presets (τα οποία είναι συμβατά με αυτά του WinFF), οι επιλογές για την ονομασία και αποθήκευση των αρχείων και οι αναδρομικές μετατροπές. Οι διαθέσιμες επιλογές στις μετατροπές video είναι τα bitrate, frame rate, η αναλογία και το μέγεθος του video, στις μετατροπές ήχου το audio bitrate, η συχνότητα και ο αριθμός καναλιών και στις μετατροπές εικόνων το μέγεθος της εικόνας. Μάλιστα, στις μετατροπές video ο χρήστης μπορεί να επεξεργαστεί απευθείας τα ορίσματα που θα περαστούν στο ffmpeg. Ακόμα, κατά τη διάρκεια της μετατροπής είναι διαθέσιμη σε πραγματικό χρόνο η έξοδος των ffmpeg/unoconv.



Όπως αναφέρθηκε, το πρόγραμμα χρησιμοποιεί το ffmpeg για τις μετατροπές ήχου/βίντεο, την βιβλιοθήκη pythonmagick για τις μετατροπές εικόνων και το unoconv για τις μετατροπές εγγράφων κειμένου. Στην πραγματικότητα, ο FF Multi Converter είναι ένα γραφικό interface για την αξιοποίηση των παραπάνω εργαλείων, το οποίο επίσης προσφέρει καλούδια όπως αυτά που αναφέρθηκαν παραπάνω.

Προς το παρόν ο FF Multi Converter είναι διαθέσιμος μόνο σε Linux.
Η εγκατάσταση είναι εύκολη και αυτή τη στιγμή υπάρχουν πακέτα για τρεις δημοφιλείς διανομές Ubuntu, Arch, Mageia.
Η εφαρμογή είναι γραμμένη σε python και PyQt4 και είναι ελεύθερο λογισμικό υπό την άδεια GNU GPLv3.

του Ιωάννη Στυλιανάκου

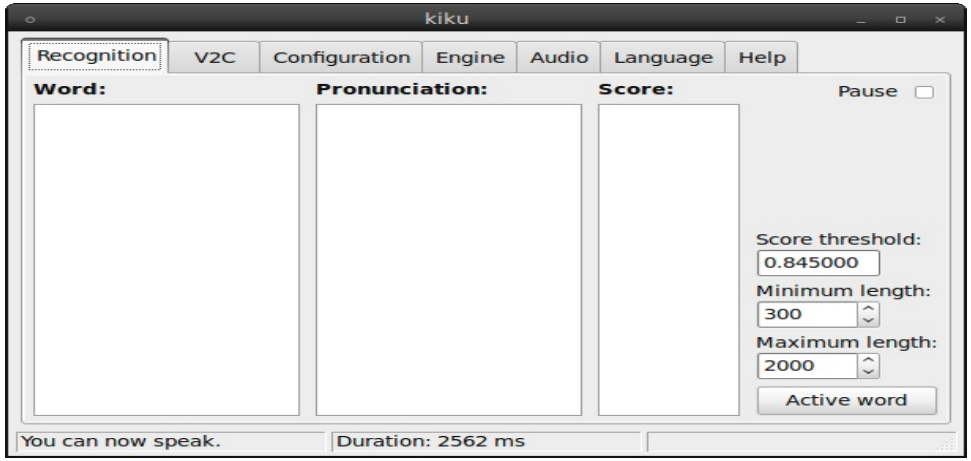
Kiku: Πρόγραμμα Αναγνώρισης Φωνής

Μιλήστε του και... θα σας καταλάβει!

Υπήρχαν στιγμές που θέλατε να μιλήσετε στον υπολογιστή σας και να σας καταλάβει και να κάνει μια εργασία για εσάς;

Υπάρχουν προγράμματα αναγνώρισης φωνής, για πλατφόρμες Windows, MacOS και Linux, με τα προγράμματα στις δύο πρώτες πλατφόρμες να σημειώνουν αρκετά καλή επιτυχία σε σχέση με αυτά του Linux. Η εφαρμογή Kiku ήρθε να ανατρέψει κάπως τα δεδομένα.

Η εφαρμογή έκανε “πρεμιέρα” έντεκα μήνες πριν, ξεκινώντας από την έκδοση 0.1 και βρίσκεται στην έκδοση 0.4. Πρόκειται για μια εφαρμογή γραμμένη σε γλώσσα C++ και χρησιμοποιεί την ανοιχτή βιβλιοθήκη ακουστικών μοντέλων αναγνώρισης φωνής Voxforge, η οποία πετυχαίνει ένα ικανοποιητικό επίπεδο λειτουργικότητας για τον αρχάριο αλλά και για τον προχωρημένο χρήστη.



Η εφαρμογή λειτουργεί ως εξής : “ακούει” από το μικρόφωνο, συγκρίνει το άκουσμα με δείγμα που έχει από την βιβλιοθήκη Voxforge, υπολογίζει την μεταξύ τους πιθανότητα (ο χρήστης μπορεί να ρυθμίσει την πιθανότητα) και στην συνέχεια εκτελεί το πρόγραμμα που έχει συσχετιστεί με τη φωνητική εντολή. Η εφαρμογή μπορεί να εκτελέσει εντολές συστήματος, προγράμματα, προσομοίωση πληκτρολογίου και ποντικιού (μέσω του xdotool). Καλή ιδέα ακούγεται ο συνδυασμός του Kiku με το πρόγραμμα σύνθεσης φωνής eSpeaker, για ένα πιο διαδραστικό περιβάλλον.

Μειονεκτήματα

- Υποστήριξη μόνο Αγγλικών και Ιαπωνικών
- Χρήση μόνο με εξωτερικό μικρόφωνο
- Χρειάζεται παραμετροποίηση

Πλεονεκτήματα

- Προσβασιμότητα στα Α.Μ.Ε.Α.
- Ευκολία χρήσης του Η/Υ
- Μικρή κατανάλωση CPU και RAM

Πηγές

http://www.workinprogress.ca/kiku/

του Γιάννη Κωστάρα

LibreOffice Writer - Εργαλεία Συνεργασίας (Μέρος 2ο)

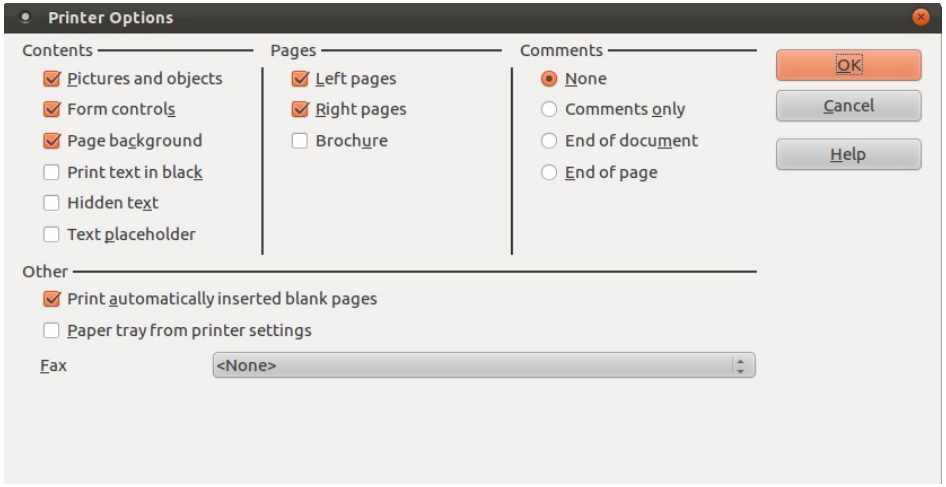
Σ' αυτό το άρθρο θα δούμε τις δυνατότητες συνεργατικής συγγραφής κειμένων μεταξύ πολλών χρηστών (collaboration) που προσφέρει το LibreOffice. Π.χ. σε μια εταιρία, ή κατά τη συγγραφή των άρθρων του περιοδικού, πολλά άτομα δουλεύουν στο ίδιο κείμενο, πάνω στο οποίο θέλουν να κάνουν αλλαγές που θέλουν να τις δουν οι υπόλοιποι ή να ανταλλάξουν απόψεις. Παρακάτω θα δούμε πώς εφαρμόζονται αυτά τα εργαλεία στην πράξη κατά τη συγγραφή ενός άρθρου στο περιοδικό Ubuntistas.

Σχόλια

Ένα ολοκληρωμένο άρθρο περνάει από επιμέλεια. Κατά την επιμέλεια του άρθρου ο επιμελητής μπορεί να χρειαστεί να προσθέσει κάποια σχόλια σε ορισμένα σημεία του άρθρου που αφορούν το συγγραφέα του άρθρου. Επιλέγει το κείμενο για το οποίο θέλει να προσθέσει σχόλια και στη συνέχεια το μενού Insert Comment. Στα δεξιά του εγγράφου θα εμφανιστεί ένα τύπου post-it πλαίσιο με το όνομα του επιμελητή και την τρέχουσα ημερομηνία στο οποίο προσθέτει το σχόλιο. Όταν ο συγγραφέας του άρθρου ανοίξει το επιμελημένο έγγραφο, μπορεί να διαγράψει το σχόλιο, ή τα σχόλια του συγκεκριμένου επιμελητή ή και όλα τα σχόλια, επιλέγοντας το κατάλληλο μενού από το δεξί κλικ πάνω στο σχόλιο.

Μπορεί να αποκρύψει την εμφάνιση σχολίων από το μενού View > Comments.

Μπορεί ακόμα να εκτυπώσει τα σχόλια μαζί με το κείμενο. Από το μενού File > Printer Settings επιλέγει το κουμπί Options και από το παράθυρο που εμφανίζεται επιλέγει πώς θέλει να εκτυπωθούν τα σχόλια.

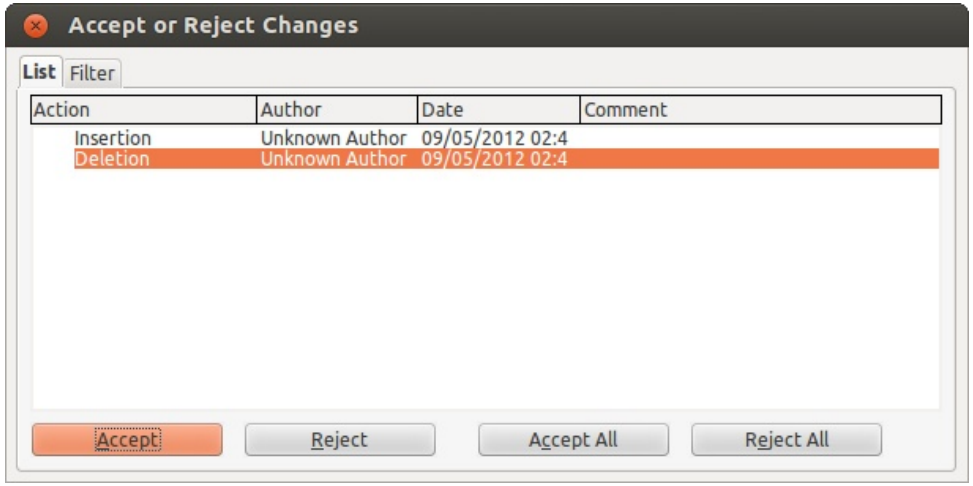


Εικόνα 1 – Printer Settings

Ανίχνευση αλλαγών

Όταν ο συγγραφέας τελειώσει με τη συγγραφή ενός άρθρου, ο επιμελητής αναλαμβάνει τη διόρθωση του άρθρου από ορθογραφικά και συντακτικά λάθη. Πριν ξεκινήσει να κάνει τις αλλαγές, ενεργοποιεί την ανίχνευση αλλαγών από το μενού Edit > Changes > Record. Για να φανούν οι αλλαγές ενεργοποιεί και την εμφάνισή τους από το μενού Edit > Changes > Show. Καθώς πληκτρολογεί βλέπει το κείμενο που εισάγει <u>υπογραμμισμένο</u> και το κείμενο που διαγράφει ως ~~διαγραμμισμένο~~. Μπορεί να ρυθμίσει πώς θα φαίνονται οι αλλαγές στο έγγραφο από το μενού Tools > Options > LibreOffice Writer > Changes. Ο επιμελητής μπορεί επίσης να προσθέσει και κάποιο σχόλιο για τη διόρθωσή του από το μενού Edit > Changes > Comment. Με αυτόν τον τρόπο, ο συγγραφέας το άρθρου μπορεί να αναθεωρήσει τις αλλαγές του επιμελητή και να τις δεχτεί ή να τις απορρίψει επιλέγοντας Edit > Changes > Accept or Reject. Εμφανίζεται το παράθυρο της εικόνας 2 όπου ο

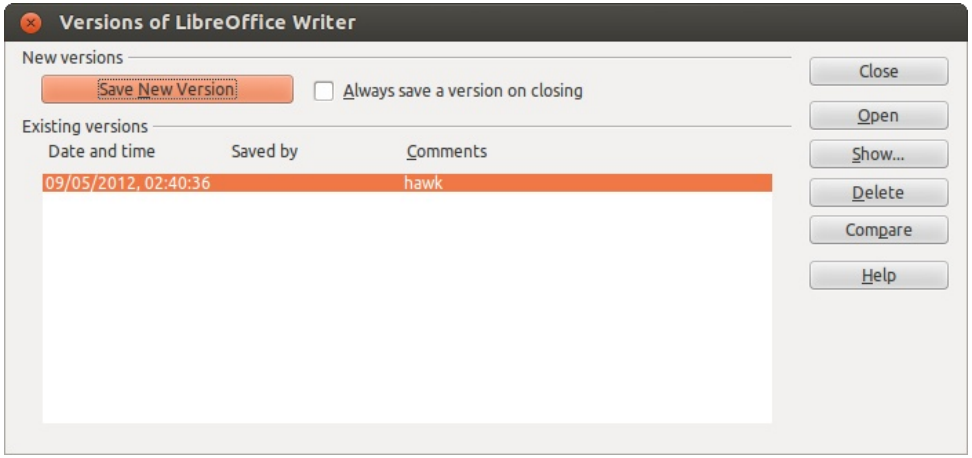
συγγραφέας μπορεί να αποδεχτεί ή να απορρίψει τις αλλαγές του επιμελητή. Ο επιμελητής μπορεί βέβαια να "κλειδώσει" τις αλλαγές του ώστε να μην μπορεί να τις πειράξει ο συγγραφέας, από το μενού Edit > Changes > Protect Record. Όταν το άρθρο περάσει και από αυτή τη φάση, είναι έτοιμο για δημοσίευση.



Εικόνα 2 – Accept or Reject track changes

Δημιουργία εκδόσεων

Ο συγγραφέας μπορεί να δημιουργήσει πολλές εκδόσεις του άρθρου δίνοντας έτσι τη δυνατότητα στον επιμελητή να ξεκινήσει τη διόρθωση χωρίς να χρειάζεται να περιμένει την ολοκλήρωση του άρθρου. Από το μενού File > Versions εμφανίζεται το παράθυρο της παρακάτω εικόνας, απ' όπου ο συγγραφέας μπορεί να δημιουργήσει μια νέα έκδοση του άρθρου του και να λάβει μια επισκόπηση των διαθέσιμων εκδόσεων. Μπορεί να ανοίξει ή να διαγράψει μια έκδοση ή ακόμα και να συγκρίνει την τρέχουσα έκδοση του εγγράφου με μια παλαιότερη. Μπορεί επίσης να επιλέξει να αποθηκεύεται αυτόματα μια νέα έκδοση με το κλείσιμο του Writer.



Εικόνα 3 – Δημιουργία εκδόσεων

Συγχώνευση εγγράφων

Μερικές φορές τυχαίνει ο συγγραφέας και ο επιμελητής να δουλεύουν σε δυο διαφορετικές εκδόσεις του ίδιου άρθρου.

Μπορείτε να αναγνωρίσετε αν κάποιος τροποποίησε το έγγραφό σας χωρίς να χρησιμοποιήσετε την αναθεώρηση ως εξής:

- Με συγχώνευση εγγράφων

- Με σύγκριση εγγράφων

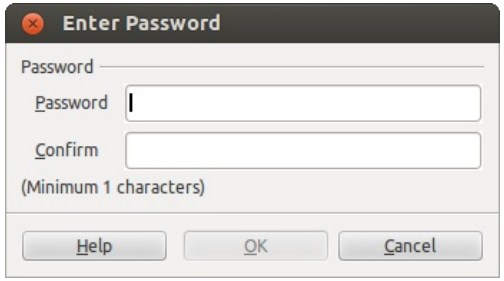
Αφού ανοίξετε το έγγραφο, κλικ στο μενού Edit > Changes > Merge Document. Επίσης μπορείτε να συγκρίνετε τα δύο έγγραφα από το μενού Edit > Compare Document.

Σε περίπτωση διένεξης, εμφανίζεται διαλογικό παράθυρο απ' όπου μπορείτε να επιλύσετε τη διένεξη.

Προστασία εγγράφου

Όταν τελειώσει και η επιμέλεια του άρθρου, καλό είναι ο συγ-

γραφέας να προστατεύει το άρθρο από ατυχή ή σκόπιμη αλλαγή του. Από το μενού File > Properties > Security ο συγγραφέας μπορεί να προστατεύσει το έγγραφο ώστε να ανοίγει μόνο για ανάγνωση καθώς και από αλλαγές (ίδιο αποτέλεσμα με το Edit > Changes > Protect Record που είδαμε παραπάνω). Μόνο όποιος γνωρίζει τον κωδικό μπορεί να ανοίξει το έγγραφο για εγγραφή και να ενεργοποιήσει την ανίχνευση αλλαγών.



Εικόνα 4 – Προστασία εγγράφου

Επίλογος

Σ' αυτό το άρθρο είδαμε τα εργαλεία που προσφέρει το LibreOffice Writer για συνεργατική συγγραφή εγγράφων. Μετά τα παραπάνω, αναμένουμε να δούμε περισσότερους συγγραφείς και επιμελητές που να θέλουν να βοηθήσουν το περιοδικό.

Πηγές:

1. Princeton University, "Collaboration Tools in Microsoft Word", http://help-desk.princeton.edu/ntfileshare/word.html.
2. LibreOffice (2011), Getting Started with LibreOffice 3.3, http://wiki.document-foundation.org/images/c/c4/0100GS3-GettingStartedLibO.pdf.
3. LibreOffice (2011), LibreOffice Writer Guide – Word Processing with LibreOffice 3.3, http://wiki.documentfoundation.org/images/b/ba/0200WG3-Writer-Guide.pdf.
4. Chanelle A. (2009), Beginning OpenOffice 3 From Novice to Professional, Apress.
5. Miller R. (2005), Point & Click OpenOffice.org!, Prentice Hall.
6. Perry E. (2011-2012), "How To – Libre Office Part 1-11", Full Circle Magazine, τεύχη 46-57.

του Γιάννη Κωστάρα

Εργαλεία Παρακολούθησης Απόδοσης Εφαρμογών

Για ν' αποδώσει μια εφαρμογή τα μέγιστα δεν αρκεί μόνο να χρησιμοποιεί πλήρως τους κύκλους μηχανής του επεξεργαστή αλλά να τους εκμεταλλεύεται κατά τέτοιο τρόπο που να μην τους σπαταλά π.χ. περιμένοντας δεδομένα από κάποια συσκευή εισόδου/εξόδου. Θα 'χετε παρατηρήσει κάποια εφαρμογή να 'κολλάει' χρησιμοποιώντας το 100% του επεξεργαστή σας χωρίς ν' αποκρίνεται, είτε γιατί περιμένει δεδομένα από κάποια μονάδα εισόδου/εξόδου είτε γιατί έπεσε σε ατέρμονα βρόχο στον κώδικά της.

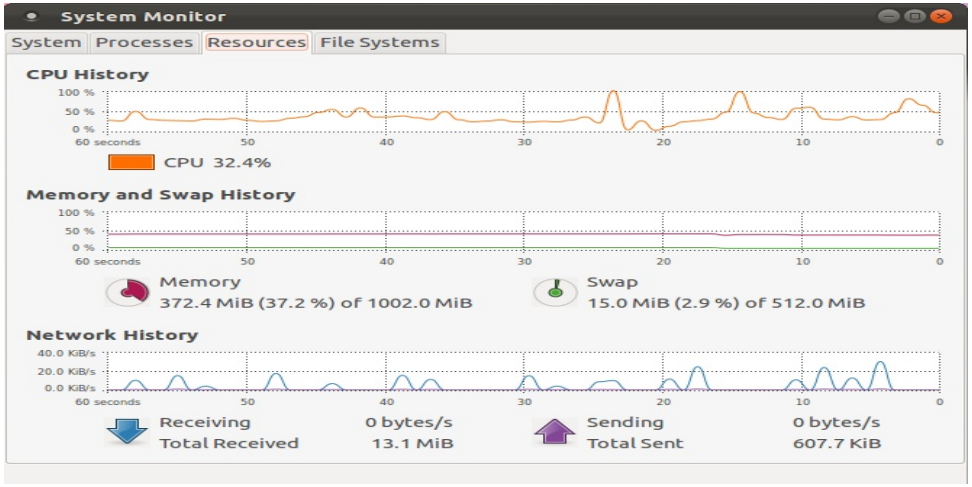
Η χρήση του επεξεργαστή ενός Η/Υ διακρίνεται σε δυο κατηγορίες: σε χρήση επεξεργασίας χρήστη/κώδικα εφαρμογής και σε χρήση επεξεργασίας πυρήνα ή συστήματος. Η πρώτη αναφέρεται στο ποσοστό χρόνου που η εφαρμογή ξοδεύει για να εκτελέσει τον κώδικά της. Η δεύτερη αναφέρεται στο ποσοστό χρόνου που η εφαρμογή ξοδεύει εκτελώντας κώδικα του λειτουργικού συστήματος για χάρη της. Όταν μια εφαρμογή ξοδεύει πολύ από το χρόνο εκτέλεσής της σε επεξεργασία πυρήνα/συστήματος, είναι μια ένδειξη σπατάλης χρόνου επεξεργασίας σε συσκευές I/O ή ανταγωνισμού (δηλ. αναμονή) ενός διαμοιραζόμενου πόρου. Η ιδανική κατάσταση θα ήταν 0% σε επεξεργασία πυρήνα/συστήματος και όσο το δυνατό μεγαλύτερο ποσοστό επεξεργασίας του κώδικα της εφαρμογής.

Στο Ubuntu η παρακολούθηση της απόδοσης του συστήματος γίνεται με διάφορα εργαλεία που διακρίνονται σε δυο κατηγορίες: γραφικά και γραμμής εντολών.

Γραφικά εργαλεία Παρακολούθησης Χρήσης Κ.Μ.Ε.

Το πιο διαδεδομένο είναι το gnome-system-monitor (βλ. Εικόνα 1):

```
$ gnome-system-monitor
```

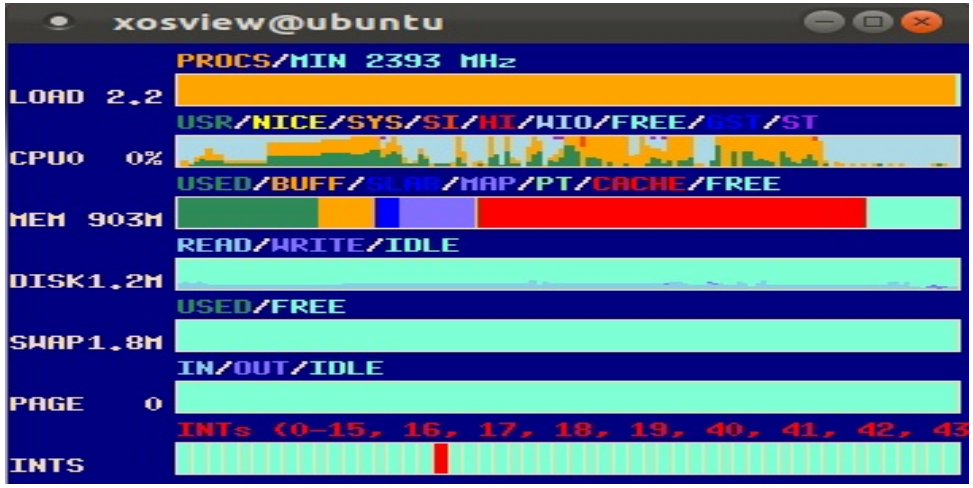


Εικόνα 1 - gnome-system-monitor

Η χρήση του επεξεργαστή φαίνεται στο ανώτερο τμήμα του παραθύρου στην καρτέλα Resources. Το σύστημά μου διαθέτει έναν μόνο επεξεργαστή, αλλά αν το δικό σας διαθέτει περισσότερους, τότε θα δείτε το φόρτο τους στο κάτω μέρος του γραφήματος. Δυστυχώς, το system monitor δεν διαχωρίζει μεταξύ χρήσης επεξεργασίας κώδικα και επεξεργασίας πυρήνα/συστήματος.

Ένα άλλο εργαλείο είναι το xosview (βλ. Εικόνα 2):

```
$ sudo apt-get install xosview
$ xosview
```



Εικόνα 2 - xosview

Αν και όχι ιδιαίτερα ευανάγνωστο, η δεύτερη γραμμή που εμφανίζει τη χρήση του επεξεργαστή διαχωρίζει ανάμεσα σε επεξεργασία χρήστη (USR) και επεξεργασία πυρήνα/συστήματος (SYS).

Μη Γραφικά Εργαλεία Παρακολούθησης Χρήσης Κ.Μ.Ε.

Πέραν των ανωτέρω γραφικών εργαλείων, υπάρχουν και εργαλεία γραμμής εντολών, όπως:

```
$ vmstat 60
```

Η εντολή αυτή εμφανίζει τη συνδυαστική χρήση όλων των εικονικών επεξεργαστών του συστήματος. Αν δε δοθεί κάποιο χρονικό διάστημα (delay) σε δευτερόλεπτα, τότε η εντολή εμφανίζει το ιστορικό από την εκκίνηση του συστήματος. Οι στήλες ενδιαφέροντος φαίνονται με έντονη γραφή (βλ. Εικόνα 3).

```
john@ubuntu:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 4  0   9184  86636  85980 481852    7   14   815   324  233  493  6 18 31 45
```

Εικόνα 3 – vmstat

Η στήλη "us" εμφανίζει το ποσοστό χρήσης επεξεργασίας της εφαρμογής. Η στήλη "sy" εμφανίζει το ποσοστό χρήσης επεξεργασίας που ξοδεύεται από το σύστημα. Η στήλη "id" εμφανίζει το ποσοστό που ο επεξεργαστής παραμένει ανενεργός. "us"+"sy"+"id" = 100%.

Οι παραπάνω πληροφορίες εμφανίζονται και σε μια πιο πινακοειδή μορφή (βλ. Εικόνα 4) με την εντολή :

```
$ sudo apt-get install sysstat
$ mpstat
```

Βλ. στήλες "%usr", "%sys" και "%idle" αντίστοιχα.

```
john@ubuntu:~$ mpstat
Linux 3.0.0-13-generic (ubuntu)         12/10/2011      _i686_  (1 CPU)

04:01:44 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest   %idle
04:01:44 PM  all    5.66    0.15   17.68   47.02    0.00    0.25    0.00    0.00   29.24
```

Εικόνα 4 - mpstat

Άλλο ένα χρήσιμο εργαλείο είναι η εντολή:

```
$ pidstat
```

που εμφανίζει πληροφορίες για κάθε διεργασία που εκτελείται (βλ. Εικόνα 5).

```
john@ubuntu:~$ pidstat
Linux 3.0.0-17-generic (ubuntu)         04/26/2012      _i686_  (1 CPU)

10:44:04 PM       PID    %usr %system  %guest    %CPU   CPU  Command
10:44:04 PM         1    0.02    1.02    0.00    1.04     0  init
10:44:04 PM         2    0.00    0.00    0.00    0.00     0  kthreadd
10:44:04 PM         3    0.00    0.01    0.00    0.01     0  ksoftirqd/0
10:44:04 PM         5    0.00    0.11    0.00    0.11     0  kworker/u:0
10:44:04 PM        15    0.00    0.00    0.00    0.00     0  khubd
10:44:04 PM        18    0.00    0.06    0.00    0.06     0  kworker/u:1
10:44:04 PM        35    0.00    0.01    0.00    0.01     0  scsi_eh_0
10:44:04 PM        36    0.00    0.01    0.00    0.01     0  scsi_eh_1
10:44:04 PM        37    0.00    0.09    0.00    0.09     0  kworker/0:2
```

Εικόνα 5 - pidstat

Τέλος, μια ακόμα χρήσιμη εντολή είναι η

```
$ top
```

η οποία εμφανίζει χρήσιμα στατιστικά στοιχεία καθώς και χρήση της μνήμης. Αποτελείται από δυο μέρη. Το άνω μέρος εμφανίζει στατιστικά στοιχεία για το σύστημα ενώ το κάτω μέρος εμφανίζει στατιστικά για κάθε διεργασία που εκτελείται απ' το σύστημα ταξινομημένες ως προς ποσοστό χρήσης της κεντρικής μονάδας επεξεργασίας.

```
top - 16:13:21 up 49 min,  1 user,  load average: 0.30, 0.64, 1.12
Tasks: 149 total,   1 running, 148 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.7%us,  1.3%sy,  0.0%ni, 97.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1026072k total,   895608k used,   130464k free,    80452k buffers
Swap:   524284k total,    14644k used,   509640k free,   442440k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 7429 root      20   0  187m  46m 9848 S  1.3  4.6   0:36.03 Xorg
 1541 tomcat6   20   0  265m  43m 3032 S  0.3  4.3   0:17.53 java
 7331 root      20   0 14460 2508 2256 S  0.3  0.2   0:09.26 vmtoolsd
 7830 john      20   0 83796  30m  10m S  0.3  3.0   0:04.93 ubuntuone-syncd
 7926 john      20   0 85188  13m  10m S  0.3  1.4   0:01.58 gnome-terminal
15781 john      20   0  2820 1164  864 R  0.3  0.1   0:00.28 top
    1 root      20   0  3444 1780 1236 S  0.0  0.2   0:03.88 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.02 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   0:00.71 ksoftirqd/0
    5 root      20   0     0    0    0 S  0.0  0.0   0:01.99 kworker/u:0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    7 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 netns
   10 root      20   0     0    0    0 S  0.0  0.0   0:00.02 sync_supers
   11 root      20   0     0    0    0 S  0.0  0.0   0:00.00 bdi-default
   12 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kintegrityd
   13 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kblockd
   14 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 ata_sff
   15 root      20   0     0    0    0 S  0.0  0.0   0:00.00 khubd
   16 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 md
   19 root      20   0     0    0    0 S  0.0  0.0   0:00.00 khungtaskd
   20 root      20   0     0    0    0 S  0.0  0.0   0:01.45 kswapd0
```

Εικόνα 6 - top

Εργαλεία Παρακολούθησης της Ουράς Εκτέλεσης του Κατανεμητή της Κ.Μ.Ε.

Ένας άλλος τρόπος παρακολούθησης του κατά πόσο το σύστημα έχει κορεστεί είναι η παρακολούθηση της ουράς εκτέλεσης του κατανεμητή (scheduler) του επεξεργαστή. Στην ουρά αυτή αποθηκεύονται διεργασίες που είναι προς εκτέλεση από τον επεξεργαστή. Αν η ουρά αυτή περιέχει πολλές διεργασίες, αυτό είναι μια ένδειξη ότι το σύστημα τείνει προς κορεσμό. Το σύστημα αρχίζει να βρίσκεται σε κορεσμό όταν το βάθος της ουράς αυτής γίνει περίπου 4 φορές ο αριθμός των εικονικών επεξεργαστών του συστήματος.

Το βάθος της ουράς εκτέλεσης είναι η πρώτη στήλη με την ετικέτα r της εντολής vmstat (βλ. Εικόνα 3). Αν το βάθος της ουράς εκτέλεσης σ' αυτό το σύστημα των 2 επεξεργαστών ξεπεράσει το

8 τότε το σύστημα υφίσταται κορεσμό και θα πρέπει να παρθούν μέτρα αντιμετώπισής του.

Εργαλεία παρακολούθησης κύριας μνήμης

Πέραν της φυσικής κύριας μνήμης, ένα σύστημα διαθέτει επίσης ιδεατή μνήμη (virtual memory) ή μνήμη σάρωσης (swap memory). Πρόκειται για το partition /swap που διατίθεται αποκλειστικά ως επέκταση της κύριας μνήμης. Όταν η κύρια μνήμη γεμίσει, τότε χρησιμοποιείται η ιδεατή μνήμη η οποία φυσικά είναι πιο αργή από την κύρια μνήμη. Το Λ.Σ. μεταφέρει μια ή περισσότερες εφαρμογές, που δεν χρησιμοποιούνται συχνά, στην ιδεατή μνήμη για να ελευθερώσει χώρο κύριας μνήμης για να εκτελέσει μια άλλη εφαρμογή. Για να εκτελεστεί μια εφαρμογή που βρίσκεται στην ιδεατή μνήμη, θα πρέπει να ξαναφορτωθεί στην κύρια μνήμη, πράγμα που έχει επίπτωση στην απόδοση της εφαρμογής. Οι στήλες free, si και so του εργαλείου vmstat (βλ. Εικόνα 3) δείχνουν την ελεύθερη μνήμη (σε KB) και τον αριθμό των φορτώσεων από και προς την ιδεατή μνήμη. Όταν ο αριθμός αυτός είναι 0, τότε δε χρησιμοποιείται η ιδεατή μνήμη. Στο σύστημα της εικόνας 3 παρατηρούμε ότι χρησιμοποιείται η ιδεατή μνήμη.

Για το σκοπό αυτό μπορεί να χρησιμοποιηθεί και η εντολή top ή επίβλεψη του αρχείου /proc/meminfo.

Εργαλεία παρακολούθησης χρήσης δικτύου

Δυστυχώς το εργαλείο netstat (μαζί με το πακέτο sysstat) δεν εμφανίζουν επαρκείς πληροφορίες για τη χρήση του δικτύου από μια εφαρμογή. Εμφανίζει μόνο στατιστικά όπως αρ. πακέτων που στάλθηκαν και λήφθηκαν ανά δευτ/πτο μαζί με λάθη και συγκρούσεις (collisions). Επιπλέον, δεν μπορούμε να γνωρίζουμε αν αυτά τα πακέτα χρησιμοποιούνται από την εφαρμογή ή όχι.

Κατεβάστε τον πηγαίο κώδικα του nicstat από: https://blogs.oracle.com/timc/entry/nicstat_the_solaris_and_linux και 'χτίστε' το κατά τα γνωστά.

```
$ nicstat [-hnsz] [-i interface[,...]] | [interval [count]]
```

όπου:

-h : εμφανίζει τη βοήθεια

-n : εμφανίζει τις μη τοπικές διεπαφές

-s : εμφανίζει μια περίληψη των αποτελεσμάτων

-z : δεν εμφανίζει μηδενικά αποτελέσματα

-i interface : εμφανίζει αποτελέσματα για τη συγκεκριμένη διεπαφή

interval : η συχνότητα εμφάνισης αποτελεσμάτων σε δευτ/πτα

count : αρ. δειγμάτων

```
john@ubuntu:~/Downloads/nicstat-1.90$ ./nicstat.sh
    Time      Int   rKB/s   wKB/s   rPk/s   wPk/s    rAvs    wAvs %Util    Sat
23:06:39       lo    0.04    0.04    0.16    0.16   255.6   255.6  0.00   0.00
23:06:39     eth0    9.78    0.60    7.95    4.39  1259.8   139.8  0.01   4.73
```

Εικόνα 7 – nicstat

```
$ sudo nicstat.sh -i eth0 1
```

Οι στήλες σημαίνουν τα εξής:

- Int : η διεπαφή δικτύου
- rKb/s : kilobytes/sec που διαβάστηκαν
- wKb/s : kilobytes/sec που γράφτηκαν
- rPk/s : αρ. πακέτων/sec που διαβάστηκαν
- wPk/s : αρ. πακέτων/sec που γράφτηκαν

- rAvs : M.O. bytes που διαβάστηκαν
- wAvs : M.O. bytes που γράφτηκαν
- %Util : χρήση του δικτύου
- Sat : τιμή κορεσμού

Από το παράδειγμα της εικόνας 8 βλέπουμε ότι η χρήση του δικτύου περιορίζεται στο 4% και δεν υπάρχει πρόβλημα κορεσμού.

```
john@ubuntu:~/Downloads/nicstat-1.90$ ./nicstat.sh -i eth0 1
    Time      Int   rKB/s   wKB/s   rPk/s   wPk/s    rAvs    wAvs %Util     Sat
23:13:16     eth0    8.20    0.50    6.68    3.69  1257.4   139.6  0.01    3.98
23:13:17     eth0    0.62    0.05    1.00    1.00   640.0    54.00  0.00    1.00
23:13:18     eth0    0.00    0.00    0.00    0.00    0.00     0.00  0.00    0.00
23:13:19     eth0    0.00    0.00    0.00    0.00    0.00     0.00  0.00    0.00
23:13:20     eth0    0.00    0.00    0.00    0.00    0.00     0.00  0.00    0.00
```

Εικόνα 8 – nicstat για το δίκτυο eth0

Εργαλεία παρακολούθησης χρήσης μονάδων εισόδου/εξόδου

Η χρήση των μονάδων εισόδου/εξόδου (π.χ. του σκληρού δίσκου) θα πρέπει επίσης να επιβλέπεται.

```
$ sudo iostat -xm
```

Το παρακάτω σύστημα εμφανίζει χρήση (%util) 26% για το δίσκο sda, με αντίστοιχη χρήση της K.M.E. (%system) 9%.

```
john@ubuntu:~$ iostat -xm
Linux 3.0.0-17-generic (ubuntu)         04/26/2012      _i686_  (1 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           8.87    2.25    8.91   14.95    0.00   65.01

Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
sda              13.05    11.79   23.85    4.82     0.37     0.18    39.28     9.76  340.21   78.10 1636.28   9.14  26.22
```

Εικόνα 9 – iostat

Άλλα εργαλεία παρακολούθησης της απόδοσης

Πέραν των παραπάνω, υπάρχουν κι άλλα εργαλεία παρακολούθησης της απόδοσης των εφαρμογών που τρέχουν σ' ένα σύστημα Linux.

Η εντολή sar χρησιμοποιείται για τη συλλογή στατιστικών απόδοσης. Μπορείτε να επιλέξετε τι είδους δεδομένα θέλετε να συλλέξετε όπως χρήση K.M.E., χρήση συστήματος ή πυρήνα, αριθμό κλήσεων συστήματος, σελιδοποίηση μνήμης, στατιστικά χρήσης δίσκου I/O κ.ά.

Το εργαλείο sar είναι μέρος του πακέτου sysstat που εγκαταστήσαμε προηγούμενα για το mpstat. Για να δουλέψει όμως, χρειάζεται μερικές ρυθμίσεις.

```
$ sudo vi /etc/default/sysstat
```

αλλάξτε τη γραμμή:

```
ENABLED="false"
```

σε

```
ENABLED="true"
```

και αποθηκεύστε τις αλλαγές.

Εξ ορισμού, το εργαλείο συλλέγει δεδομένα κάθε 10 λεπτά. Αν θέλετε να το αλλάξετε αυτό:

```
$ sudo vi /etc/cron.d/sysstat
```

και τροποποιήστε το κατάλληλα.

```
$ sudo service sysstat restart
$ sar
```

```
john@ubuntu:~$ sar
Linux 3.2.0-24-generic (ubuntu)         05/06/2012      _i686_  (1 CPU)

09:59:34 AM       LINUX RESTART

10:05:01 AM     CPU     %user     %nice   %system   %iowait    %steal     %idle
10:15:02 AM     all      6.17      2.78      5.96     15.86      0.00     69.24
Average:        all      6.17      2.78      5.96     15.86      0.00     69.24

07:29:27 PM       LINUX RESTART

07:30:42 PM       LINUX RESTART

07:35:02 PM     CPU     %user     %nice   %system   %iowait    %steal     %idle
07:45:01 PM     all      6.26      9.03     10.94     11.71      0.00     62.05
07:55:01 PM     all      1.54      0.00      1.59      1.38      0.00     95.49
Average:        all      3.88      4.47      6.22      6.49      0.00     78.94
```

Εικόνα 10 – sar

Αν θέλετε να δείτε όλα τα στατιστικά:

```
$ sar -A
```

Αν θέλετε να αποθηκεύσετε τα στατιστικά σ' ένα αρχείο:

```
$ sudo sar -A > $(date +`hostname`-%d-%m-%y-%H%M.log)
```

Ανατρέξτε στις αναφορές για περισσότερες πληροφορίες σχετικά με το εργαλείο.

Επίλογος

Σ' αυτό το άρθρο είδαμε διάφορα εργαλεία που μπορούν να χρησιμοποιηθούν από το διαχειριστή ενός συστήματος ή από έναν ειδικό μέτρησης της απόδοσης εφαρμογών ή ακόμα και από κάποιον προγραμματιστή για να μετρήσουν την απόδοση μιας ή περισσοτέρων εφαρμογών. Είδαμε εργαλεία μέτρησης της χρήσης της Κ.Μ.Ε., της κύριας μνήμης και των μονάδων εισόδου/εξόδου (δικτύου και δίσκων) από μια εφαρμογή.

Μάθαμε για τις δυο κατηγορίες στις οποίες διαχωρίζεται η επεξεργασία μιας εφαρμογής: τη χρήση επεξεργασίας χρήστη/κώδικα εφαρμογής και τη χρήση επεξεργασίας πυρήνα ή συστήματος και μιλήσαμε για τα διάφορα εργαλεία που διαθέτει το Ubuntu για την παρακολούθηση της απόδοσης εφαρμογών. Πλέον διαθέτετε ένα ολόκληρο “οπλοστάσιο” από εργαλεία για να μπορείτε να βρίσκετε πού ξοδεύονται οι πόροι του συστήματός σας.

Πηγές:

1. Hunt C. & Binu J. (2011), Java Performance, O' Reilly.

2. “How to configure sysstat/sar on Ubuntu/Debian”, http://www.leonardoborda.com/blog/how-to-configure-sysstatsar-on-ubuntudebian/

3. “10 Useful Sar (Sysstat) Examples for UNIX / Linux Performance Monitoring”, http://www.thegeekstuff.com/2011/03/sar-examples/

4. “SYSSTAT Howto: A Deployment and Configuration Guide for Linux Servers”, https://www.linux.com/learn/tutorials/33766-sysstat-howto-a-deployment-and-configuration-guide-for-linux-servers

του Γιάννη Κωστάρα

Erlang

Η Erlang, ή Ericsson Language, αναπτύχθηκε από την Ericsson για χρήση στις τηλεπικοινωνίες ήδη από τα μέσα της δεκαετίας του 1980 όταν καμία άλλη γλώσσα δεν κάλυπτε τις ανάγκες τους. Το 1999 δημοσιεύθηκε ως ανοικτού κώδικα και ιδρύθηκε το http://erlang.org.

Πρόκειται για μια συναρτησιακή γλώσσα προγραμματισμού (βλ. LISP για τους παλαιότερους) όπως π.χ. οι Scala, Clojure, Haskell, OCaml κ.ά. σε αντίθεση με τις imperative γλώσσες προγραμματισμού όπως οι C/C++, Pascal, Java, Python, Ruby κλπ. Τι σημαίνει αυτό;

Στις συναρτησιακές γλώσσες προγραμματισμού, οι συναρτήσεις επιστρέφουν το αποτέλεσμα εφαρμόζοντας πράξεις στα ορίσματά τους χωρίς να αλλάζουν την τιμή τους. Π.χ. στη Java αντικείμενα που η κατάστασή τους δεν αλλάζει μετά τη δημιουργία τους λέγονται αμετάβλητα (immutable). Σε αντίθεση, οι διαδικαστικές (procedural) ή imperative γλώσσες προγραμματισμού εφαρμόζουν πράξεις στα ορίσματά τους αλλάζοντας όμως την κατάστασή τους. Η παραπάνω ιδιότητα των συναρτησιακών γλωσσών προγραμματισμού έχει πολύ μεγάλα πλεονεκτήματα όσον αφορά τον πολυδιεργασιακό προγραμματισμό (multi-threading) καθώς δεν απαιτούν συγχρονισμό αφού δεν γίνεται διαμοιρασμός δεδομένων και άρα δεν υπάρχουν κρίσιμα τμήματα κώδικα. Επομένως, τμήματα κώδικα μπορούν να εκτελεστούν ταυτόχρονα από πολλά νήματα (threads) ή διεργασίες (processes) χωρίς πρόβλημα.

Στην Erlang εντολές του τύπου i++ ή i=i+1 δεν επιτρέπονται. Με άλλα λόγια, δεν επιτρέπεται η αλλαγή της τιμής μιας μεταβλητής μόλις αυτή οριστεί για πρώτη φορά.

Η Erlang είναι επίσης μια δηλωτική (declarative) γλώσσα προγραμματισμού, όπως θα δούμε στη συνέχεια, δηλ. περιγράφεις τι θέλεις να υπολογίσει κι όχι πως να το υπολογίσει.

Επίσης, οι συναρτήσεις της (ή closures) είναι κι αυτές τύποι δεδομένων που μπορούν να εκχωρηθούν σε μεταβλητές!

Αν ακόμα αναρωτιέστε γιατί γράφω για αυτήν τη σχετικά άγνωστη γλώσσα, θα αναφερθώ σε μερικές εφαρμογές που είναι γραμμένες σε Erlang:

- CouchDB, μια document-based ΒΔ που χρησιμοποιεί το MapReduce

- SimpleDB, μια κατανεμημένη ΒΔ που χρησιμοποιείται από τα Amazon Web Services

- η εφαρμογή συνομιλιών (chat) του facebook

- Wings 3D, μια 3Δ εφαρμογή μοντελοποίησης

- RabbitMQ, μια υλοποίηση του Advanced Message Queuing Protocol (AMQP)

Εγκατάσταση

Από το Ubuntu Software Center, ψάξτε για erlang και θα βρείτε το πακέτο “Concurrent, real-time, distributed functional language” για εγκατάσταση.

Αν θέλετε όμως την τελευταία έκδοση, τότε κατεβάστε τον πηγαίο κώδικα από τη διεύθυνση http://www.erlang.org/download.html και δώστε τις ακόλουθες εντολές μέσα στον κατάλογο που αποσυμπιέσατε τον πηγαίο κώδικα για να τον χτίσετε:

```
$ ./configure
$ make
$ sudo make install
```

Αν όλα πήγαν καλά, πρέπει να είστε έτοιμοι να ξεκινήσετε. Δια-

φορετικά, ελέγξτε το αρχείο INSTALL.md της διανομής που κατεβάσατε για τυχόν προβλήματα.

Το περιβάλλον εργασίας

Προσδεθείτε λοιπόν, είμαστε έτοιμοι να ξεκινήσουμε! Ανοίξτε ένα κέλυφος και δώστε την εντολή:

```
$ erl
Erlang R13B03 (erts-5.7.4) [source] [rq:1] [async-threads:0]
[hipe] [kernel-poll:false]

Eshell V5.7.4  (abort with ^G)
1>q().
ok
2> $
```

Αυτό είναι το αλληλεπιδραστικό περιβάλλον εργασίας. Η έκδοσή σας μπορεί να διαφέρει από την παραπάνω.

Κάθε εντολή της Erlang τελειώνει με την τελεία (.). Για να βγείτε από το περιβάλλον εργασίας δώστε την εντολή q(). όπως φαίνεται παραπάνω.

Ο συντάκτης του κελύφους βασίζεται στον Emacs, οπότε όσοι έχετε δουλέψει μ' αυτόν θα βρείτε αρκετές συντομεύσεις χρήσιμες.

Αριθμοί

Η γλώσσα υποστηρίζει τις βασικές αριθμητικές πράξεις, δηλ. +, -, *, /, div (ακέραια διαίρεση), rem (ακέραιο υπόλοιπο) ενώ δεν υπάρχει διαφορά μεταξύ ακεραίων και δεκαδικών. Ισχύουν φυσικά τα γνωστά για την προτεραιότητα των πράξεων.

```
1> 5 + 2 * (15 div 4).
11
2> 2 * 3.14.
6.28
```

Η αναπαράσταση αριθμών σε βάση διαφορετική του δεκαδικού γίνεται με τη σύνταξη <βάση>#<αριθμός>:

```
3> 2#101.
5
4> 16#F.
15
```

Πειραματιστείτε με μερικούς πραγματικά μεγάλους αριθμούς για να δείτε πόσο γρήγορα εμφανίζονται τ' αποτελέσματα. Στην Erlang δεν υπάρχει υπερχείλιση.

Η σύνταξη $a επιστρέφει τον ASCII κωδικό του a (δηλ. 97).

Σχόλια

```
5> % This is a comment
5>
```

Σταθερές

Οι σταθερές στην Erlang ονομάζονται Atoms και μπορούν να είναι αλφαριθμητικά, αριθμοί κλπ. Ξεκινούν με μικρό γράμμα, ενώ αν θέλετε σώνει και καλά να ορίσετε μια σταθερά που να ξεκινά με κεφαλαίο γράμμα, ορίστε τη μέσα σε απλά ή διπλά εισαγωγικά, π.χ.:

```
5> atom.
atom
6> 'Atom'.
```

```
Atom
7> atom = 'bla'.
** exception error: no match of right hand side value bla
```

Όπως βλέπετε από την εντολή 7, μια σταθερά δεν μπορεί να λάβει άλλη τιμή. Από πλευράς απόδοσης, αποθηκεύονται σ' έναν πίνακα συστήματος ως 2 bytes ανεξάρτητα απ' το μέγεθός τους.

Σημειώστε ότι οι παρακάτω είναι δεσμευμένες λέξεις και επομένως δεν μπορούν να χρησιμοποιηθούν ως σταθερές:

after and andalso band begin bnot bor bsl bsr bxor case catch cond div end fun if let not of or orelse query receive rem try when xor.

Μεταβλητές

Οι μεταβλητές στην Erlang δηλώνονται με κεφαλαίο το πρώτο γράμμα:

```
8> Var.
* 1: variable 'Var' is unbound
9> Var = 1.
1
10> Var = 2.
** exception error: no match of right hand side value 2
11> Var = Var + 1.
** exception error: no match of right hand side value 2
```

Οι παραπάνω εντολές μας δείχνουν ότι οι μεταβλητές στην Erlang είναι αμετάβλητες (immutable) δηλ. μπορούμε να εκχωρήσουμε μια τιμή σε μια μεταβλητή μόνο μια φορά. Αν προσπαθήσουμε να εκχωρήσουμε μια διαφορετική τιμή σε μεταβλητή επί της οποίας έχει ήδη εκχωρηθεί τιμή, τότε θα εμφανιστεί λάθος.

Η εξήγηση είναι ότι ο τελεστής = παίζει το ρόλο της ταύτισης προτύπων (pattern matching) κι όχι της εκχώρησης, όπως σε άλλες γλώσσες προγραμματισμού, και άρα παραπονιέται όταν η τιμή αριστερά και δεξιά του δεν είναι ίσες. Έτσι η έκφραση X = X + 1 είναι λάθος όπως ακριβώς μας έμαθε κι ο δάσκαλος των μαθηματικών. Επομένως, Pattern = Expression εκτιμά την έκφραση Expression και προσπαθεί να ταιριάξει το αποτέλεσμα με το πρότυπο Pattern. (Άμα θέλετε να χρησιμοποιήσετε την ίδια μεταβλητή στο περιβάλλον εργασίας, δώστε την εντολή f(). ώστε να ξεχάσει τις προηγούμενες εκχωρήσεις).

Το γεγονός ότι οι μεταβλητές στην Erlang είναι αμετάβλητες έχει τεράστια σημασία, όπως θα δούμε παρακάτω, στη συγγραφή πολυδιεργασιακών προγραμμάτων, καθώς δεν υπάρχει διαμοιραζόμενη μνήμη (shared memory) και τα επακόλουθα προβλήματα συγχρονισμού της, όπως σε άλλες γλώσσες (βλ. Java, C++, Ruby κλπ.)

Η Erlang είναι δυναμική γλώσσα, όπως η Ruby, δηλ. δεν χρειάζεται να δηλώσετε τον τύπο δεδομένων της μεταβλητής σας· τον καταλαβαίνει απ' τα συμφραζόμενα.

Υπάρχουν φυσικά και λογικές σταθερές και λογικοί τελεστές (σημειώστε ότι τα true και false είναι atoms):

```
12> true and false.
false
13> false or true.
true
14> true xor false.
true
15> not false.
true
```

Υπάρχουν οι ακόλουθοι τελεστές ισότητας:

```
16> 5 =:= 5.
true
17> 1 =:= 0.
false
```

```
18> 1 =/= 0. % άνισο
true
19> 5 =:= 5.0.
false
20> 5 == 5.0.
true
21> 5 /= 5.0.       % άνισο
false
```

και οι γνωστοί τελεστές σύγκρισης <, >, <=, >=.

```
22> 0 == false.
false
23> 1 < false.
true
```

Μάλλον θα τραβάτε τα μαλλιά σας απ' τις δυο τελευταίες εντολές. Σε πολλές γλώσσες, 0 == false και 1 == true, όχι όμως στην Erlang. Η αλήθεια είναι ότι τα true και false είναι atoms. Και ισχύει η ακόλουθη σειρά μεταξύ τύπων δεδομένων:

number < atom < reference < fun < port < pid < tuple < list < bit string

Θα μιλήσουμε για τους τύπους αυτούς στη συνέχεια του άρθρου.

Πλειάδες (Tuples)

Μια πλειάδα στην Erlang δηλώνεται ως {Στοιχείο1, Στοιχείο2, ..., ΣτοιχείοΝ} όπου κάθε στοιχείο μπορεί να είναι διαφορετικού τύπου. Η διαφορά τους από τις λίστες, που θα δούμε στη συνέχεια, είναι ότι οι πλειάδες έχουν σταθερό μήκος, δηλ. δεν μπορούμε να προσθέσουμε ή ν' αφαιρέσουμε στοιχεία απ' αυτές αφότου τις δηλώσουμε.

```
24> Point = {0.0, 5.0}.
{0.0,5.0}
25> {X,Y} = Point.
{0.0,5.0}
26> X.
0.0
```

Θα μπορούσαμε να αντιστοιχίσουμε την πλειάδα με έναν πίνακα κατακερματισμού (HashMap) όπου τα κλειδιά είναι atoms π.χ.

```
27> {points, {p1,{10.0,0.0}}, {p2,{0.0,5.0}}}.
{points,{p1,{10.0,0.0}},{p2,{0.0,5.0}}}
```

Πώς όμως μπορούμε να προσπελάσουμε τα στοιχεία της πλειάδας; Όπως μόλις είδαμε, χωρίς να το αναφέρουμε, στη γραμμή 25 πιο πάνω, με τη χρήση pattern matching.

```
28> Points={points,{p1,{10.0,0.0}}, {p2,{0.0,5.0}}}.
29> {points, {p1,{P1X,P1Y}}, {p2,{_,_}}} = Points.
30> P1X.
10.0
```

Καθώς ένα atom (π.χ. points, p1) ταιριάζει με τον εαυτό του, το μόνο που μένει είναι να ταιριάξει τις μεταβλητές P1X, P1Y. Το _ απλά δηλώνει ότι δεν μας ενδιαφέρουν οι αντίστοιχες τιμές.

Όταν το πρώτο στοιχείο μιας πλειάδας είναι ένα atom, λέμε ότι αποτελεί την ετικέτα της πλειάδας, π.χ. {person, 'Yiannis', 'Kostaras'}.

Λίστες

Οι λίστες είναι οι πιο χρήσιμες δομές δεδομένων στις συναρτησιακές γλώσσες καθώς χρησιμοποιούνται για να επιλύσουν πολλά είδη προβλημάτων. Οι λίστες μπορούν να περιλαμβάνουν τα πάντα: αριθμούς, σταθερές, πλειάδες κλπ. Η σύνταξή της είναι [Στοιχείο1, Στοιχείο2, ..., ΣτοιχείοΝ] και όπως ειπώθηκε, κάθε Στοιχείο μπορεί να ανήκει σε διαφορετικό τύπο δεδομένων:

```
27> [1, 2, 3, {point,[5.0,-6.0]}, 3.14, atom].
[1,2,3,{point,[5.0,-6.0]},3.14,atom]
28> [97, 98, 99].
"abc"
```

Η τελευταία εντολή πρέπει να σας μπέρδεψε τελείως. Όπως βλέπετε, τους διαχειρίζεται ως ASCII κωδικούς αντί για αριθμούς. Αυτό είναι ένα θέμα με τη γλώσσα. Αν της πείτε να τυπώσει μόνο αριθμούς που αντιστοιχούν σε γράμματα, τότε τυπώνει τα γράμματα γιατί τους μεταφράζει σε κωδικούς ASCII. Κι αυτό γιατί στην πραγματικότητα δεν υπάρχουν αλφαριθμητικά στην Erlang· τα αλφαριθμητικά είναι λίστες ακεραίων αριθμών! Δηλαδή το αλφαριθμητικό "abc" είναι στην ουσία η λίστα [97, 98, 99]. Με άλλα λόγια, αν θέλετε να κάνετε επεξεργασία αλφαριθμητικών, τότε καλύτερα να απευθυνθείτε σε άλλες γλώσσες προγραμματισμού που τα καταφέρνουν καλύτερα σ' αυτό τον τομέα όπως Perl, Python, Ruby κ.ά.

Υποστηρίζονται οι παρακάτω πράξεις: ++, -- οι οποίες εκτελούνται από δεξιά προς τ' αριστερά.

```
29> [1,2,3] ++ [4,5].
[1,2,3,4,5]
30> [1,2,3,4,5] -- [1,2,3].
[4,5]
```

Το πρώτο στοιχείο μιας λίστας ονομάζεται Κεφαλή (Head) και τα υπόλοιπα στοιχεία Ουρά (Tail):

```
31> hd([1,2,3,4,5]).
1
32> tl([1,2,3,4,5]).
[2,3,4,5]
33> [Head|Tail] = [1,2,3,4,5].
[1,2,3,4,5]
34> Head.
1
35> Tail.
[2,3,4,5]
36> [3 | []].
[3]
37> [2 | [3 | []]].
[2,3]
38> [1 | [2 | [3 | []] ] ].
[1,2,3]
39> [1 | [2, 3] ].
[1,2,3]
```

Ο τελεστής | ονομάζεται cons (constructor – κατασκευαστής/δημιουργός) και χρησιμοποιείται για να δημιουργεί μια λίστα από κεφαλή και ουρά.

Το module lists (θα μιλήσουμε για την έννοια του module παρακάτω) παρέχει μια πληθώρα από συναρτήσεις πάνω στις λίστες:

```
40> lists:max([1,2,3]).
3
41> lists:reverse([1,2,3]).
[3,2,1]
42> lists:sort([2,1,3]).
[1,2,3]
43> lists:split(2,[1,2,3,4]).
{[1,2],[3,4]}
44> lists:sum([1,2,3]).
6
45> lists:zip([1,2,3],[5,6,7]).
[{1,5},{2,6},{3,7}]
46> lists:delete(2,[1,2,3,2,4,2]).
[1,3,2,4,2]
47> lists:last([1,2,3]).
3
48> lists:member(5,[1,2,3]).
false
49> lists:nth(2,[3,4,10,7,9]).
4
50> length([1,2,3]).
3
```

Παρατηρήστε ότι η length είναι ενσωματωμένη στις βιβλιοθήκες συναρτήσεων της γλώσσας οι οποίες λέγονται Build-In

Functions (ή BIFs). Άλλες BIFs είναι οι hd και tl που είδαμε πιο πάνω.

Δομές ελέγχου

Η γλώσσα υποστηρίζει δυο δομές ελέγχου ροής, την case και την if.

```
if
  Guard1 ->
    Sequence1 ;
  Guard2 ->
    Sequence2 ;
  ...
end

51> X = 10.
10
52> if
52>   X > 0 -> 2*X;
52>   X < 0 -> X/2;
52>   true -> 0
52> end.
20.
```

Η τελευταία συνθήκη θα μπορούσε να γραφτεί και ως X==0 -> 0.

Η case συντάσσεται ως εξής:

```
case Expr of
  Pattern1 [when Guard1] -> Seq1;
  Pattern2 [when Guard2] -> Seq2;
  ...
  PatternN [when GuardN] -> SeqN
end
```

```
53> Greeting = "morning".
"morning"
54> case Greeting of
54>  morning -> "good morning";
54>  afternoon -> "good afternoon";
54>  evening -> "good evening";
54>  _ -> "good day"
54> end.
"good morning"
```

Το underscore (_) παίζει το ρόλο του default σε άλλες γλώσσες προγραμματισμού.

Συναρτήσεις

Οι συναρτήσεις στην Erlang ορίζονται μέσα σε modules. Η Erlang διαθέτει ένα σύστημα τμηματοποίησης που μας επιτρέπει να χωρίζουμε μεγάλα προγράμματα σε μικρότερα για ευκολότερη διαχείριση. Κάθε τέτοιο τμήμα (module) έχει το δικό του χώρο ονομασίας (name space) οπότε μπορούμε να χρησιμοποιούμε τα ίδια ονόματα συναρτήσεων με άλλων τμημάτων χωρίς πρόβλημα. Τα modules επομένως, είναι κάτι αντίστοιχο με τα πακέτα (packages) της Java ή των χώρων ονομασίας (namespaces) της C++, και αποθηκεύονται σε αρχεία με κατάληξη .erl. Ας δούμε ένα παράδειγμα. Αντιγράψτε το ακόλουθο πρόγραμμα στον αγαπημένο σας κειμενογράφο και αποθηκεύστε το σ' ένα αρχείο με όνομα myecho.erl:

```
-module(myecho).
-export([echo/1]).
echo(Message) ->
  Message.
```

Το παραπάνω module ονομάζεται myecho και αποθηκεύεται στο αρχείο myecho.erl. Η διεπαφή του με τον έξω κόσμο δηλώνεται με την εντολή export η οποία δηλώνει μια συνάρτηση που εί-

ναι διαθέσιμη στον έξω κόσμο που ονομάζεται echo και δέχεται μια μόνο παράμετρο (/1). Στη συνέχεια ακολουθεί η υλοποίηση της συνάρτησης η οποία δέχεται το όρισμά της στη μεταβλητή Message (προσέξτε το πρώτο γράμμα να είναι κεφαλαίο) και απλά σαν έξοδο επιστρέφει το περιεχόμενο της Message. Μπορείτε να ορίσετε κι άλλες συναρτήσεις σ' αυτό το module αλλά από τη στιγμή που δεν τις δηλώσετε στην export αυτές είναι εσωτερικές στο module.

Μεταφερθείτε στον κατάλογο που αποθηκεύσατε το αρχείο σας και μεταγλωττίστε το:

```
55> cd("/home/john/erlang/").
/home/john/erlang/
ok
56> c(myecho).
{ok,myecho}
57> myecho:echo(dirlada).
dirlada
```

Το αποτέλεσμα της μεταγλώττισης είναι η δημιουργία ενός αρχείου .beam. Η εκτέλεσή του γίνεται παραθέτοντας <το όνομα του module> : <το όνομα της συνάρτησης με τα ορίσματά της>.

Ας δούμε και το παρακάτω πρόγραμμα μετατροπής θερμοκρασιών από Fahrenheit σε Celsius και αντίστροφα:

```
-module(temperature).
-export([convert/2]).

convert({fahrenheit, Temp}, celsius) ->
  {celsius, 5 * (Temp - 32) / 9};
convert({celsius, Temp}, fahrenheit) ->
  {fahrenheit, 32 + Temp * 9 / 5};
convert({X, _}, Y) ->
  {cannot,convert,X,to,Y}.
```

Μεταγλωττίστε το κατά τα γνωστά:

```
58> c(temperature).
{ok,temperature}
59> temperature:convert({celsius, 23}, fahrenheit).
{fahrenheit,73.4}
```

Παρατηρήστε πως δουλεύει το pattern matching:

```
{celsius, Temp}, fahrenheit = {celsius, 23}, fahrenheit
Temp <-> 23
```

και πως έτσι επιλέγεται το πρώτο μέρος της συνάρτησης convert.

Αναδρομή

Όπως γνωρίζετε, μια συνάρτηση (υποπρόγραμμα) μπορεί να κληθεί από μια άλλη συνάρτηση αλλά ακόμα και από την ίδια συνάρτηση. Στη δεύτερη περίπτωση η συνάρτηση λέγεται αναδρομική γιατί καλεί το εαυτό της. Η αναδρομή πολλές φορές απλοποιεί την επίλυση δύσκολων προβλημάτων με λίγες μόνο εντολές. Στην Erlang οι αναδρομικές συναρτήσεις προκύπτουν από το συνδυασμό ταιριάσματος προτύπων (pattern matching) και συναρτήσεων.

Ας δούμε ένα παράδειγμα, τον υπολογισμό του παραγοντικού ενός φυσικού αριθμού N:

- Αν N=0 τότε N! = 1

- Αν N>0 τότε N! = N*(N-1)!

Από τα παραπάνω βλέπουμε ότι για να υπολογίσουμε το παραγοντικό του N θα πρέπει πρώτα να υπολογίσουμε το παραγοντικό του N-1. Για να υπολογίσουμε το παραγοντικό του N-1 χρειάζεται πρώτα να υπολογίσουμε το παραγοντικό του N-2 κ.ο.κ. μέχρις ότου φθάσουμε σε μια συνθήκη όπου μπορούμε να υπολογίσουμε το παραγοντικό, η οποία λέγεται συνθήκη διακοπής, και στο παραπάνω παράδειγμα είναι το 0!=1. Σημειώστε ότι χωρίς συνθήκη διακοπής το πρόγραμμα μπαίνει σε ατέρμονα βρόγχο.

Δημιουργήστε ένα νέο αρχείο recursion.erl:

```
-module(recursion).
```

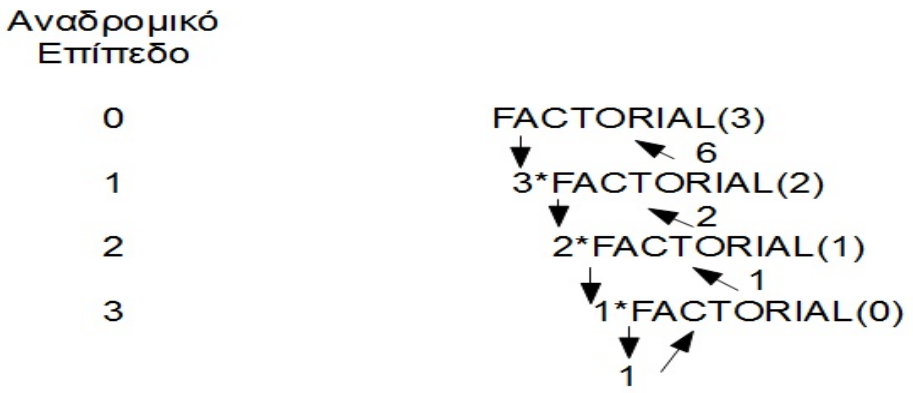
```
-export([factorial/1]).
factorial(0) ->
    1;
factorial(N) ->
    N * factorial(N - 1).
```

Κατ' αρχήν κοιτάξτε πόσο μικρό είναι το πρόγραμμα και πως αντιστοιχίζεται με τη θεωρία. Επίσης παρατηρήστε ότι η συνάρτηση factorial αποτελείται από δυο μέρη που χωρίζονται με ;. Αν το όρισμα που θα την καλέσουμε είναι 0, τότε καλείται ο πρώτος ορισμός, διαφορετικά γίνεται ταίριασμα με τον δεύτερο ορισμό και υπολογίζεται το παραγοντικό αναδρομικά.

```
60> cd("/home/john/erlang/").
/home/john/erlang/
ok
61> c(recursion).
{ok,recursion}
62> recursion:factorial(3).
6
```

Ας δούμε πως δουλεύει (βλ. Σχήμα 1).

Σχήμα 1 – Παρουσίαση της εκτέλεσης της αναδρομής



Κάθε νέα κλήση της factorial αυξάνει το αναδρομικό επίπεδο. Ο μεταγλωττιστής αποθηκεύει τις τιμές της παραμέτρου N της συνάρτησης όπως επίσης και οποιασδήποτε τοπικής μεταβλητής πριν από κάθε νέα κλήση. Κατά τη διάρκεια της κλήσης υπολογίζεται η έκφραση N-1 και καταχωρείται ως η νέα τιμή της παραμέτρου N. Τελικά, η παράμετρος N θα γίνει 0 και θα δοθεί η τιμή 1 ως αποτέλεσμα της factorial (προς τα πίσω διαδικασία). Στη συνέχεια, ο μεταγλωττιστής επιστρέφει από κάθε αναδρομικό επίπεδο και υπολογίζει την τρέχουσα τιμή της factorial πολλαπλασιάζοντας την παράμετρο που έχει αποθηκεύσει σ' αυτό το επίπεδο με την τιμή της factorial του προηγούμενου επιπέδου. Κατ' αυτόν τον τρόπο επιστρέφουμε στο αρχικό επίπεδο (προς τα εμπρός διαδικασία). Στο σχήμα 1 υπάρχουν 4 αναδρομικά επίπεδα (0-3) και 3 αναδρομικές κλήσεις της factorial συν την αρχική κλήση.

Αυτό που εντυπωσιάζει ότι το πόσο γρήγορα υπολογίζει η Erlang το παραγοντικό μεγάλων αριθμών. Συγκρίνετέ τη με τον υπολογισμό του παραγοντικού από γλώσσες όπως η Java ή η Ruby.

Οι αριθμοί fibonacci είναι άλλο ένα κλασικό παράδειγμα αναδρομής:

- F(0) = 0
- F(1) = 1
- Αν N>1 τότε F(N) = F(N-1) + F(N-2)

Τροποποιήστε το πρόγραμμα recursion ως εξής:

```
-module(recursion).
-export([factorial/1]).
-export([fib/1]).

% N!
factorial(0) ->
    1;
factorial(N) ->
```

```
    N * factorial(N - 1).

% fib(N) = fib(N-1) + fib(N-2)
fib(0) -> 0;
fib(1) -> 1;
fib(N) -> fib(N-1) + fib(N-2).

63> c(recursion).
{ok,recursion}
64> recursion:fib(12).
144
```

Για λίγο μεγαλύτερους αριθμούς όμως η αναδρομή δεν αποτελεί τον πιο αποτελεσματικό τρόπο επίλυσης ενός προβλήματος όπως μπορείτε να πειραματιστείτε με το παραπάνω πρόγραμμα. Συνήθως, αλγόριθμοι με χρήση βρόγχων είναι πιο γρήγοροι. Παρόλ' αυτά ο παρακάτω αλγόριθμος υπολογισμού των αριθμών Fibonacci αποδεικνύεται ικανοποιητικά γρήγορος:

- F(0) = 0
- F(1) = 1
- Αν N>1 τότε:
- F(2N) = 2F(N-1)*F(N) + F(N)*F(N)
- F(2N-1) = F(N)*F(N) + F(N-1)*F(N-1)

Προσθέστε ακόμα μια συνάρτηση στο module recursion, την fastfib και προσθέστε τον κώδικά της στο τέλος του αρχείου:

```
fastfib(0) -> 0;
fastfib(1) -> 1;
% F(2N) = 2F(N-1)*F(N) + F(N)*F(N)
fastfib(N) when N rem 2 == 0 ->
      FN = fastfib(N div 2),
      2*FN*fastfib(N div 2-1) + FN*FN;
% F(2N+1) = F(N)*F(N) + F(N+1)*F(N+1)
% F(2N-1) = F(N)*F(N) + F(N-1)*F(N-1)
fastfib(N) ->
      FN = fastfib((N-1) div 2),
      FN_1 = fastfib((N+1) div 2),
      FN*FN + FN_1*FN_1.
```

Παρατηρήστε τη συνθήκη φρουρό fastfib(N) when N rem 2 == 0 όταν ο N είναι ζυγός αριθμός για να ξεχωρίσουμε πότε να χρησιμοποιηθεί το ένα σκέλος της συνάρτησης και πότε το άλλο καθώς και τα δυο δέχονται ίδιο αριθμό παραμέτρων. Μπορούμε να έχουμε πολλές συνθήκες σ' έναν φρουρό οι οποίες μπορούν να διαχωρίζονται με κόμμα (,) που σημαίνει ότι θα επιστρέψει true αν όλες οι συνθήκες επιστρέψουν true (σύζευξη), ή με ελληνικό ερωτηματικό (;) που θα επιστρέψει true αν οποιαδήποτε συνθήκη επιστρέψει true (διάζευξη).

Επίσης, μέσα στο ίδιο μπλοκ οι εντολές χωρίζονται με κόμμα (,).

```
65> c(recursion).
{ok,recursion}
66> recursion:fib(45).
1134903170
```

Μπορούμε να συγκρίνουμε τους δυο αλγορίθμους με τη χρήση της συνάρτησης συστήματος timer:tc(Module, Function, Arguments) η οποία επιστρέφει μια πλειάδα με το χρόνο σε microseconds και το αποτέλεσμα της συνάρτησης:

```
67> timer:tc(recursion, fib, [45]).
{208335434,1134903170}
68> timer:tc(recursion, fastfib, [45]).
{21,1134903170}
```

Στο μηχάνημά μου, η fib πήρε 208 δευτερόλεπτα για να εκτελεστεί ενώ η fastfib 21 μsecs!!!

Ως άσκηση για τον αναγνώστη αφήνεται η επίλυση του παρακάτω αναδρομικού προβλήματος:

- S(1) = 1

- Αν N>1 τότε S(N) = N + S(N-1)

το οποίο υπολογίζει το άθροισμα: 1+2+...N.

Γράψτε τις παραπάνω συναρτήσεις σε C ή Java και συγκρίνετε πόσες παραπάνω γραμμές κώδικα χρειάζεστε (για να μη μιλήσω για απόδοση)!

Πολυεπεξεργασία

Ίσως το πιο δυνατό σημείο της γλώσσας είναι οι δυνατότητες πολυεπεξεργασίας της. Η διαφορά της εικονικής μηχανής της Erlang από άλλες εικονικές μηχανές είναι ότι η πολυεπεξεργασία παρέχεται από την ίδια την εικονική μηχανή κι όχι από το λειτουργικό σύστημα. Έτσι, μας δίνεται η δυνατότητα να δημιουργήσουμε χιλιάδες ή ακόμα και εκατομμύρια διεργασίες που να εκτελούνται παράλληλα χωρίς πρόβλημα. Οι διεργασίες επικοινωνούν μεταξύ τους μέσω ασύγχρονων μηνυμάτων.

Η πολυεπεξεργασία γίνεται με τις ακόλουθες εντολές:

- spawn: ξεκινά μια διεργασία

- (!) send: στέλνει ένα μήνυμα σε μια διεργασία

- receive: λαμβάνει ένα μήνυμα από μια διεργασία.

Μπορείτε να δημιουργήσετε μια διεργασία με την εντολή spawn(Module, Function, Parameters) περνώντας της ως ορίσματα το module, τη συνάρτηση και τυχόν ορίσματα της συνάρτησης με τη μορφή λίστας, π.χ.:

```
Pid = spawn(recursion, fastfib, [N]).
```

και μας επιστρέφει το PID ή Process ID της διεργασίας που δημιούργησε.

Οι διεργασίες επικοινωνούν μεταξύ τους με μηνύματα. Π.χ.

```
69> Pid=self().
<0.33.0>
70> Pid ! hello.
hello
71> Pid ! "hello again".
"hello again"
72> flush().
Shell got hello
Shell got "hello again"
ok
```

Η BIF self() μας επιστρέφει το pid της τρέχουσας διεργασίας που δεν είναι άλλη από το κέλυφος της Erlang. Σ' αυτήν στέλνουμε το μήνυμα hello και το hello again. Τα μηνύματα αυτά καταλήγουν στο γραμματοκιβώτιο του κελύφους. Η εντολή flush() καθαρίζει το γραμματοκιβώτιο.

Η λήψη και επεξεργασία των μηνυμάτων γίνονται με τη μέθοδο receive() που περιέχει εντολές σε στυλ case:

```
receive
  Pattern1 [when Guard1] -> Seq1;
  Pattern2 [when Guard2] -> Seq2;
  ...
  PatternN [when GuardN] -> SeqN
end
```

Η receive είναι μια μέθοδος η οποία 'ακούει' για μηνύματα. Κάθε διεργασία έχει τη δική της ουρά λήψης μηνυμάτων. Κάθε νέο μήνυμα εισάγεται στο τέλος της ουράς. Όταν εκτελείται η receive, το πρώτο μήνυμα της ουράς προσπαθεί να ταυτιστεί με κάποιο από τα πρότυπα της receive, κι όταν συμβεί κάτι τέτοιο τότε απομακρύνεται από την ουρά και εκτελείται, διαφορετικά σε περίπτωση μη ταιριάσματος με κανένα πρότυπο παραμένει στην ουρά και το επόμενο μήνυμα ακολουθεί την ίδια διαδικασία.

Ας δούμε ένα παράδειγμα ενός echo server ο οποίος επιστρέφει

στον καλούντα το μήνυμα που του απέστειλε.

```
-module(echo).
-export([start/0, loop/0]).
start() ->
  spawn(echo, loop, []).
loop() ->
  receive
   {From, Message} ->
       From ! {reply, Message},
     io:format("Received: ~p, Sent: ~p~n", [Message, Message]),
       loop()
  end.
```

Η μέθοδος start() ξεκινά μια νέα διεργασία με την εντολή spawn(Module, Function, Parameters) και επιστρέφει το process id (ή pid) της όπως φαίνεται παρακάτω. Αυτή η διεργασία-παιδί ξεκινά να εκτελείται στη συνάρτηση loop/0 και πηγαίνει σε αναμονή με το που φθάνει στην εντολή receive καθώς το γραμματοκιβώτιό της είναι άδειο.

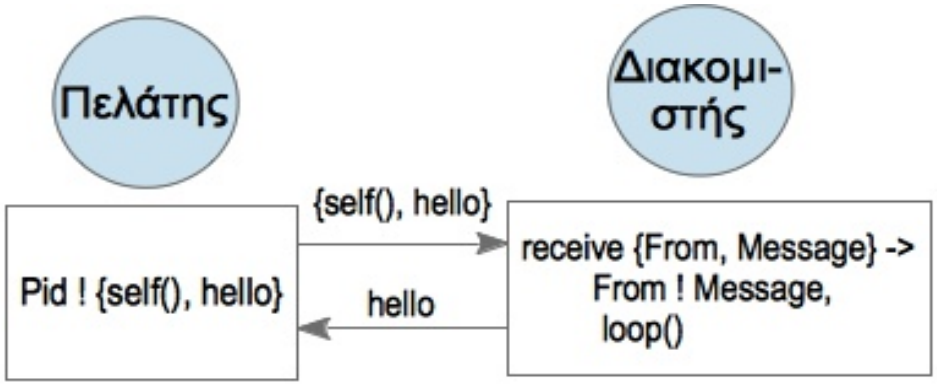
```
73> c(echo).
{ok,echo}
74> Pid = echo:start().
<0.38.0>
75> Pid ! {self(), hello}.
Received: hello, Sent: hello
{<0.31.0>,hello}
```

Για να στείλουμε ένα μήνυμα σ' αυτή τη διεργασία, χρησιμοποιούμε την πιο πάνω εντολή (75). Στη διεργασία με pid Pid (δηλ. την <0.38.0>) στέλνουμε την πλειάδα {self(), hello} δηλ. το pid της τρέχουσας διεργασίας (που στην περίπτωσή μας είναι το κέλυφος erl) και το μήνυμα hello.

Η receive ακούει συνεχώς για μηνύματα. Η παραπάνω πλειάδα ταυτίζεται με τη {From, Message} <-> {self(), hello} η οποία απλά στέλνει το Message (hello) πίσω στην τρέχουσα διεργασία (From <-> self()) υπό τη μορφή πλειάδας {reply, Message}. Στην περίπτωσή μας, το κέλυφος λαμβάνει το μήνυμα και τυπώνει {<0.31.0>,hello}, όπου <0.31.0> είναι το pid του κελύφους.

Η loop() καλεί τη receive για να ελέγξει αν ήρθε κάποιο μήνυμα η οποία ξανακαλεί τη loop() αναδρομικά αναμένοντας στη receive για το επόμενο μήνυμα.

Σχήμα 2 – Απεικόνιση πολυεπεξεργασίας echo server



Συνήθως καταχωρούμε ψευδώνυμα στα pids με την BIF register(Alias, Pid) και εργαζόμαστε μ' αυτά.

```
76> register(echo, echo:start()).
true
77> whereis(echo).
<0.42.0>
78> echo ! {self(), hello}.
Received: hello, Sent: hello
{<0.31.0>,hello}
```

Ας δούμε ακόμα ένα παράδειγμα, το πρόγραμμα fibonacci.

```
-module(concur_fib).
-export([start/0, loop/0]).
-import(recursion, [fastfib/1]).
start() ->
```

```
  spawn(concur_fib, loop, []).
loop() ->
  receive
   {From, N} ->
       F = fastfib(N),
       From ! {reply, F},
       io:format("Received: ~p, Replied: ~p~n", [N, F]),
       loop()
  end.
```

Η import είναι αντίστροφη της export και εισάγει συναρτήσεις από άλλα modules (στη συγκεκριμένη περίπτωση την fastfib του module recursion). Από εκεί και πέρα το πρόγραμμα δεν κρύβει εκπλήξεις.

```
79> f().
ok
80> c(concur_fib).
{ok,concur_fib}
81> Pid = concur_fib:start().
<0.94.0>
82> Pid ! {self(), 50}.
Received: 50, Replied: 12586269025
{<0.94.0>,50}
```

Όπως καταλαβαίνετε, μπορείτε να ξεκινήσετε πολλές διεργασίες που θα υπολογίζουν τους αριθμούς fibonacci. Επίσης, η Erlang σας επιτρέπει να διαμοιράσετε τις διεργασίες σας σε διαφορετικούς υπολογιστές (βλ. π.χ. [1-6]).

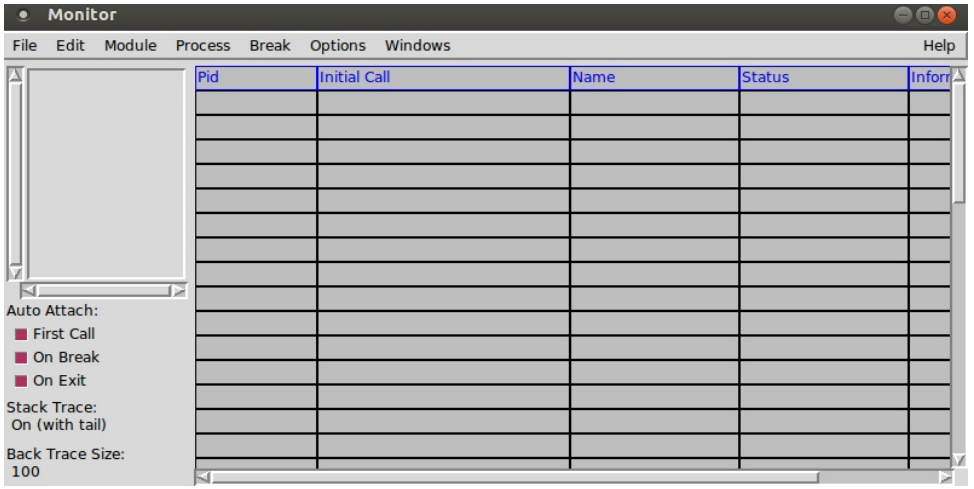
Μπορείτε να δείτε ποιες διεργασίες τρέχουν στην εικονική μηχανή σας δίνοντας την εντολή processes(). Οι εντολές i() και regs() εμφανίζουν ποια συνάρτηση εκτελεί κάθε διεργασία.

Ολοκληρωμένα Περιβάλλοντα Εργασίας

Δυστυχώς, δεν υπάρχουν. Η γλώσσα υποστηρίζεται αφού κατεβάσετε το κατάλληλο plugin για το Eclipse (erlide) και το Intellij IDEA. Για το NetBeans, το διαθέσιμο plugin, ErlyBird, έχει σταματήσει να αναπτύσσεται από το 2009, οπότε αν κάποιος από εσάς έχει την όρεξη και το χρόνο ας με βοηθήσει να το προχωρήσουμε.

Δεν πρέπει να παραλείψουμε και τον γραφικό αποσφαλματωτή που έρχεται μαζί με το περιβάλλον της γλώσσας:

```
83> debugger:start().
```



Εικόνα 3 – Erlang debugger

Το module σας θα πρέπει να μεταγλωττιστεί με την ένδειξη debug_info. Αυτό μπορεί να επιτευχθεί με δυο τρόπους: είτε από ένα κέλυφος με την εντολή:

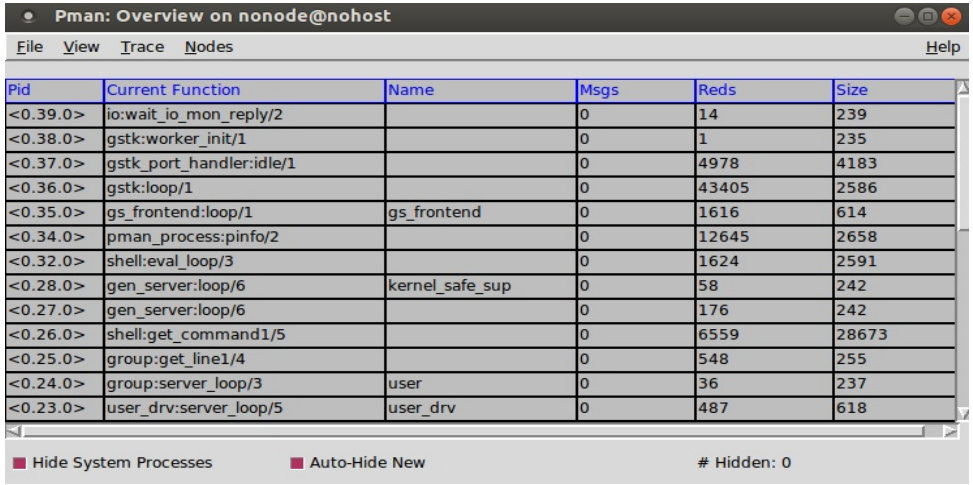
```
$ erlc +debug_info Module.erl
```

είτε από το περιβάλλον της Erlang:

```
1> c(Module, [debug_info]).
2> compile:file(exception, [debug_info]).
```

Επιλέξτε το μενού Module > Interpret και φορτώστε το erl αρχείο. Όταν εκτελέσετε το πρόγραμμά σας αυτό θα εμφανιστεί στον debugger. Κάντε διπλό κλικ πάνω στο Pid του για εμφανιστεί ένα παράθυρο απ' όπου μπορείτε να δείτε τις τιμές των μεταβλητών και να εκτελέσετε το πρόγραμμα γραμμή-γραμμή.

Για την αποσφαλμάτωση των παράλληλων προγραμμάτων, υπάρχει ο Process Manager (βλ. Εικόνα 4).



Εικόνα 4 – Process Manager

Για να τον εκκινήσετε, στο κέλυφος δώστε:

```
3> pman:start().
```

Η έξοδος είναι παρόμοια με αυτή της εντολής i(). Κάντε διπλό κλικ σε μια διεργασία για ν' ανοίξετε το παράθυρο ανίχνευσης όπου μπορείτε να ανιχνεύσετε όλα τα μηνύματα που έστειλε/έλαβε η διεργασία.

Άσκηση

Γράψτε σε Erlang το παιχνίδι πέτρα-ψαλίδι-χαρτί. Κάθε παίχτης επιλέγει να παίξει πέτρα (P), ψαλίδι (S), χαρτί (X). Η πέτρα νικάει το ψαλίδι, το ψαλίδι νικάει το χαρτί και το χαρτί νικάει την πέτρα.

```
Είσοδος: [["Giannis","S"], ["Kostas","P" ]]
Έξοδος: ["Kostas","P"] wins since P>S
```

Δημιουργήστε μια διεργασία που να καλεί τη συνάρτηση αυτή και να της επιστρέφει τ' αποτελέσματα του υπολογισμού της.

Επίλογος

Σ' αυτό το άρθρο δώσαμε μια εισαγωγή στο συναρτησιακό προγραμματισμό με τη σχετικά άγνωστη συναρτησιακή γλώσσα προγραμματισμού πραγματικού χρόνου Erlang. Γνωρίσαμε το συντακτικό της καθώς και τις δυνατότητές της για πολυεπεξεργασία (concurrency) που είναι και το πιο δυνατό της στοιχείο. Βέβαια, στις λίγες σελίδες αυτού του άρθρου είναι αδύνατο να καλύψουμε όλες τις δυνατότητες τις γλώσσας, όπως π.χ. η διαχείριση λαθών, οι βιβλιοθήκες OTP, κ.ά. αλλά πλέον έχετε τις βάσεις για να ανατρέξετε στη βιβλιογραφία για περαιτέρω μελέτη.

Αξίζει στο σημείο αυτό να αναφέρουμε το έργο ErLLVM [11] του ΕΜΠ για τη χρήση μιας Χαμηλού Επιπέδου Εικονικής Μηχανής - Low Level Virtual Machine (LLVM) προς μια Erlang Υψηλής Απόδοσης – High Performance Erlang (HiPE).

Είτε χρειάζεστε μια γλώσσα για να εκτελέσετε επιστημονικούς υπολογισμούς, είτε μια γλώσσα για να εκμεταλλευτείτε όλους τους επεξεργαστές του υπερσύγχρονου συστήματός σας, η Erlang είναι μια πολύ ενδιαφέρουσα και πρωτότυπη γλώσσα που μπορεί να σας λύσει τα χέρια με πολύ λίγες γραμμές κώδικα.

Πηγές:

1. Getting started with Erlang User's Guide, v. 5.9 (2011), http://www.erlang.org/doc/getting_started/users_guide.html.
2. Erlang Reference Manual User's Guide, v. 5.9 (2011), http://www.erlang.org/doc/reference_manual/users_guide.html.
3. Learn you some Erlang, http://learnyousomeerlang.com/.
4. Armstrong J. (2007), Programming Erlang, Pragmatic.
5. Cesarini F. & Thompson S. (2009), Erlang Programming, O'Reilly.
http://en.wikibooks.org/wiki/Erlang_Programming.
6. 3 Free E-Books and a Tutorial on Erlang, http://www.readwriteweb.com/hack/2011/05/free-e-books-on-erlang.php.
7. Tate B. (2010), Seven languages in seven weeks: A Pragmatic guide to learning programming languages, Pragmatic.
8. Fibonacci calculator, http://www.tools4noobs.com/online_tools/fibonacci/.
9. Fast Fibonacci algorithms, http://nayuki.eigenstate.org/page/fast-fibonacci-algorithms.
10. ErLVVM, http://erllvm.softlab.ntua.gr/.

του Ιωάννη Στυλιανάκου

Foremost και επαναφορά δεδομένων

Γνωρίζετε το συναίσθημα που δημιουργείται όταν διαγραφεί ένα σύνολο σημαντικών αρχείων που πραγματικά είναι απαραίτητα. Είναι ένα φοβερό συναίσθημα και συχνά ακολουθείται από ανησυχία, πανικό και επανάληψη του "γιατί να το κάνω αυτό;". Μην πανικοβάλλεστε· υπάρχει ελπίδα για τα διαγραμμένα αρχεία, ακόμη και αν το μέσο μαζικής αποθήκευσης έχει μορφοποιηθεί. Το Foremost μπορεί επίσης να ανακτήσει κατεστραμμένα αρχεία. Είναι κυρίως ένα εργαλείο ανάκτησης δεδομένων το οποίο αρχικά γράφτηκε από τους Kris Kendall και Jesse Kornblum, ειδικοί πράκτορες για το Γραφείο Ειδικών Ερευνών της Πολεμικής Αεροπορίας των Ηνωμένων Πολιτειών. Πάρθηκε και τροποποιήθηκε από τον Nick Mikus ως μέρος της διατριβής του και είναι τώρα διαθέσιμο και στο Ubuntu. Με το πρόγραμμα στα αποθετήρια, η εγκατάστασή του είναι ένα απλό θέμα:

Ανοίγουμε τερματικό και δίνουμε:

```
$ sudo apt-get install foremost
```

Προσοχή πριν προχωρήσετε περαιτέρω - ΜΗΝ προσαρμόσετε ή εκκινήσετε τη μονάδα που θέλετε να ανακτήσετε. Όσο περισσότερο προσβάσιμο είναι το μέσο τόσο μεγαλύτερη είναι η πιθανότητα απώλειας δεδομένων. Πριν προχωρήσουμε στην ανάκτηση των δεδομένων, κάνουμε ένα αντίγραφο ασφαλείας του αρχικού μέσου. Μία από τις βασικές αρχές των δεδομένων ανάκτησης είναι να εργαστείτε με ένα αντίγραφο του αρχικού μέσου και όχι το ίδιο το μέσο. Μία άλλη σημαντική αρχή είναι να δημιουργήσετε αντίγραφα ασφαλείας σε ένα μέσο που βρίσκεται στο ίδιο φυσικό μέσο με αυτό που θέλετε να κάνετε επαναφορά (προφανώς δε θέλετε να αλλοιώσετε το δίσκο ενώ προσπαθείτε να ανακτήσετε τα στοιχεία από αυτόν!). Η μονάδα δίσκου αντιγράφων ασφαλείας θα πρέπει να έχει αρκετό ελεύθερο χώρο για να χωρέσει μια εικόνα ολόκληρου του δίσκου (με τα χαμένα αρχεία). Σε αυτό το παράδειγμα, θα γίνει ανάκτηση δεδομένων από μια μονάδα flash 1GB σε ένα σύστημα με σκληρό 80GB.

Ξεκινάμε κάνοντας ένα αντίγραφο του δίσκου που θέλουμε να επαναφέρουμε:

```
$ sudo dd if=/dev/sdb1 of=mypendrive.img
```

Στη συνέχεια, πρέπει να δώσουμε στο χρήστη κυριότητα του αρχείου που μόλις δημιουργήσαμε. Σε αυτή την περίπτωση, το χρήστη και την ομάδα με το όνομα charly:

```
$ sudo chown charly:charly mypendrive.img
```

Το Foremost ανακτά πολλούς διαφορετικούς τύπους δεδομένων. Ένα άλλο εργαλείο, το Photorec (μέρος του πακέτου TestDisk), στην πραγματικότητα αναγνωρίζει πολλά περισσότερα, αλλά το Foremost μπορεί να λειτουργήσει με μη προσαρτημένες μονάδες και με εικονικά αρχεία. Κυρίως χρειάζεται μια διαδρομή για να να αποθηκεύσετε τα δεδομένα. Αυτή η διαδρομή δεν θα πρέπει να είναι στο μέσο που θέλετε να επαναφέρετε.

```
$ mkdir ~/recovery
```

Τώρα, ας ανακτήσουμε κάποια pdf και png αρχεία:

```
$ foremost -vqQ -o recovery/ -t pdf,png -i mypendrive.img
```

Ο διακόπτης -v δίνει τη δυνατότητα στο Foremost για λειτουργία σε verbose mode. Χωρίς το διακόπτη -v, το Foremost παρουσιάζει αστερίσκους "*" στην οθόνη κατά την διαδικασία ανάκτησης.

Ο διακόπτης -v μας δίνει ωραία διαμορφωμένο αποτέλεσμα :

```
441: 00702752.png 233 KB 359809024 (800 x 480)
442: 00703392.png 177 KB 360136704 (1024 x 640)
443: 00703776.png 239 KB 360333312 (640 x 360)
```

Πηγές

1. *Ubuntu Help: http://tinyurl.com/7chjpmw*
2. *Ubuntu Geek: http://tinyurl.com/4t3qhq*
3. *Ubuntu man pages: http://tinyurl.com/ccjvs6q*

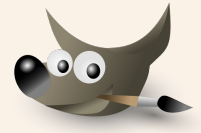
Ubuntistas
Το περιοδικό της Ελληνικής Κοινότητας Ubuntu-gr
ubuntistas.ubuntu-gr.org

Το ubuntistas σε χρειάζεται!

Για να μπορέσει να συνεχίσει να λειτουργεί το περιοδικό μας, όπως καταλαβαίνετε, χρειάζεται συνεχώς νέα άρθρα. Αν έχεις καμιά ιδέα και θες να συνεισφέρεις γράφοντας άρθρα μπες στο **http://ubuntistas.ubuntu-gr.org/index.php/contact** και επικοινώνησε μαζί μας!

Για την κατασκευή-σχεδιασμο του ubuntistas χρησιμοποιήθηκαν

LibreOffice
The Document Foundation



ubuntu-gr