ΑΤΕΙ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

Τμήμα Μηχανικών Πληροφορικής ΤΕ

Ρυθμίσεις Samba με γραφικό περιβάλλον

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Βασίλης Κούλης (ΑΜ: Τ03197)

Επιβλέπων: ***<Χαϊκάλης Κωνσταντίνος, τίτλος, βαθμίδα>***

ΛΑΡΙΣΑ 2014

«Εγώ ο/η Βασίλης Κούλης, δηλώνω υπεύθυνα ότι η παρούσα Πτυχιακή Εργασία με τίτλο Ρυθμίσεις Samba με γραφικό περιβάλλον είναι δική μου και βεβαιώνω ότι:

- *Σε όσες περιπτώσεις έχω συμβουλευτεί δημοσιευμένη εργασία τρίτων, αυτό επισημαίνεται με σχετική αναφορά στα επίμαχα σημεία.*
- *Σε όσες περιπτώσεις μεταφέρω λόγια τρίτων, αυτό επισημαίνεται με σχετική αναφορά στα επίμαχα σημεία. Με εξαίρεση τέτοιες περιπτώσεις, το υπόλοιπο κείμενο της πτυχιακής αποτελεί δική μου δουλειά.*
- *Αναφέρω ρητά όλες τις πηγές βοήθειας που χρησιμοποίησα.*
- *Σε περιπτώσεις που τμήματα της παρούσας πτυχιακής έγιναν από κοινού με τρίτους, αναφέρω ρητά ποια είναι η δική μου συνεισφορά και ποια των τρίτων.*
- *Γνωρίζω πως η λογοκλοπή αποτελεί σοβαρότατο παράπτωμα και είμαι ενήμερος(-η) για την επέλευση των νομίμων συνεπειών»*

< υπογραφή >

< ονοματεπώνυμο >

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Τόπος: .....

Ημερομηνία: .....

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. .....
2. .....
3. .....

Περίληψη

Στην εργασία αυτή θα ασχοληθούμε με την εφαρμογή Samba η οποία χρησιμοποιείται ευρέως για διαμοιρασμό αρχείων σε τοπικό δίκτυο ανάμεσα σε υπολογιστές Windows και Linux. Θα δούμε την ιστορία του Samba και του GTK+ όπως και τις δυνατότητές τους. Επίσης θα δούμε κάποιες εφαρμογές οι οποίες χρησιμοποιούν αυτά τα εργαλεία για διαμοιρασμό μέσω Samba και τέλος θα αναλύσουμε την εφαρμογή που δημιουργήθηκε στην παρούσα εργασία. Θα δούμε πως λειτουργεί, όπως και τον κώδικά της, τί κάνει η κάθε κλάση και τί η κάθε μέθοδος.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω για τη συμβολή τους στην τελική διαμόρφωση του κειμένου τον αδερφό μου, Παναγιώτη και τον φίλο μου Τάσο Μακρή.

Βασίλης Κούλης

ημερομηνία

Περιεχόμενα

ΠΕΡΙΛΗΨΗ ..... **I**

ΕΥΧΑΡΙΣΤΙΕΣ ..... **III**

ΠΕΡΙΕΧΟΜΕΝΑ ..... **V**

1 ΕΙΣΑΓΩΓΗ **1**

2 ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ **3**

2.1 SAMBA 3

2.1.1 Ιστορία *3*

2.1.2 Πρωτόκολλα *5*

2.1.3 Τρόπος λειτουργίας διαμοιρασμού μέσω Samba *7*

2.2 GTK+ 9

2.2.1 Ιστορία *9*

2.2.2 Τρόπος Λειτουργίας *9*

2.2.3 Που χρησιμοποιείται *9*

2.2.4 Επεξήγηση Ονομασιών [4] *10*

3 ΑΝΑΛΥΣΗ ΑΝΤΙΣΤΟΙΧΩΝ ΕΦΑΡΜΟΓΩΝ **11**

3.1 GADMIN-SAMBA 11

3.2 SYSTEM CONFIG SAMBA 13

4 ΠΡΩΤΑ ΒΗΜΑΤΑ **15**

4.1 ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ 16

4.1.1 Settings Tab *16*

4.1.2 Shares Tab *17*

4.1.3 Users Tab *20*

4.1.4 Configuration Tab *21*

5 ΚΩΔΙΚΑΣ **23**

5.1 ΕΙΣΑΓΩΓΗ 23

5.2 CLASS WINDOW 24

5.3 CLASS WINFUNC 28
5.4 CLASS BUILDER 42
6 ΣΥΜΠΕΡΑΣΜΑΤΑ 45
ΒΙΒΛΙΟΓΡΑΦΙΑ 47

1 Εισαγωγή

Το κίνητρο για να ασχοληθώ με τη συγκεκριμένη πτυχιακή εργασία αποτέλεσε το γεγονός ότι χρησιμοποιώ καθημερινά το Samba στο δίκτυό στο σπίτι και είχα ανάγκη από έναν απλό τρόπο για να μοιράζομαι εύκολα τους φακέλους του υπολογιστή μου. Αν και υπάρχουν κάποιες εφαρμογές από τρίτους, τους λείπουν όμως κάποιες δυνατότητες, όπως διαμοιρασμός του εκτυπωτή, ενώ ταυτόχρονα θεωρώ τις εν λόγω εφαρμογές δύσχρηστες. Στην εφαρμογή που ανέπτυξα χρειάζεται να υπάρχουν δικαιώματα διαχειριστή, κι επομένως στην αρχή πρέπει να εισάγουμε τον αντίστοιχο κωδικό.

2 Θεωρητικό Υπόβαθρο

2.1 Samba

Το Samba[1,2] είναι ένα πρόγραμμα που δημιουργήθηκε από τον Andrew Tridgell το οποίο είναι φτιαγμένο για Linux, Apple και άλλα βασισμένα στο Unix συστήματα. Είναι προεγκατεστημένο στις περισσότερες εκδόσεις Linux και η άδειά του είναι σε GNU. Επικοινωνεί με τα πρότυπα SMB και CIFS τα οποία χρησιμοποιούν τα Windows για να μοιράζονται αρχεία σε τοπικό δίκτυο.

2.1.1 Ιστορία

Έκδοση 0.1 έως 1.0 (Δεκέμβρη 1991)

Ο δημιουργός του άρχισε την ανάπτυξη του προγράμματος στο πανεπιστήμιο της Αυστραλίας χρησιμοποιώντας Packet Sniffing για να δει πως λειτουργεί το πρωτόκολλο μεταφοράς αρχείων του προγράμματος Pathworks. Σε αυτές τις εκδόσεις δεν είχε βρει κάποιο επίσημο όνομα και απλά το ονόμαζε «Unix file server for Dos Pathworks». Στην έκδοση 1.0 ανακάλυψε ότι μπορεί να χρησιμοποιηθεί και σε άλλους προσωπικούς υπολογιστές.

Έκδοση 1.5 (Δεκέμβρη 1993)

Τότε συνεργαζόμενος με την Microsoft έβγαλε την καινούρια έκδοση η οποία περιείχε και client και server. Σε αυτή την έκδοση διάλεξε και την άδεια να είναι GPL2 όπως και το όνομα. Στην αρχή είχε διαλέξει το όνομα smbserver, αλλά επειδή ειδοποιήθηκε ότι το όνομα χρησιμοποιείται ήδη, το άλλαξε σε Samba.

Έκδοση 1.6 έως 2.2.0 (Ιανουάριο 1995)

Τότε έβγαλε την έκδοση 1.6 όπου σε αυτήν την έκδοση άρχισαν οι συνεισφορές και έτσι ξεκίνησε η ομάδα Samba τον Μάιο του 1996.

Έκδοση 3.0.x (Σεπτέμβριο 2003)

Τότε βγήκε μια μεγάλη αναβάθμιση στην οποία υπάρχει η δυνατότητα να συνδέεται σε Domain. Σε αυτές τις εκδόσεις απέκτησε και κάποιες καινούριες λειτουργείες. Η τελευταία έκδοση αυτής της σειράς ήταν η 3.0.37 η οποία βγήκε τον Οκτώβριο του 2009.

Έκδοση 3.1

Αυτή η έκδοση χρησιμοποιήθηκε μόνο για την ανάπτυξη του προγράμματος.

Έκδοση 3.2 (Ιούλιο 2008)

Τότε αποφάσισαν να βγάζουν εκδόσεις κάθε 6 μήνες οι οποίες θα είχαν διορθώσεις προβλημάτων. Σε αυτή την έκδοση η άδεια άλλαξε από GPL2 σε GPL3 και μερικά κομμάτια σε LGPL3.

Έκδοση 3.3 (Ιανουάριο 2009)

Η τελευταία έκδοση ήταν η 3.3.16.

Έκδοση 3.4 (Ιούλιο 2009)

Ήταν η πρώτη έκδοση που περιείχε κώδικα και από το Samba 3 και από το Samba 4. Η τελευταία έκδοση που βγήκε ήταν η 3.4.17 η οποία ήταν η πιο σταθερή έκδοση μέχρι τις αρχές του 2014.

Έκδοση 3.5 (Μάρτιο 2010)

Ήταν η πρώτη έκδοση που βγήκε που είχε πειραματική υποστήριξη για το SMB2

Έκδοση 3.6 (Αύγουστος 2011)

Σε αυτή την έκδοση είχε πλήρη υποστήριξη του SMB2. Επίσης από την 3.6.3 έκδοση και μετά αυξήθηκε και η ασφάλεια διορθώνοντας αρκετά προβλήματα που υπήρχαν.

Έκδοση 4.x (Δεκέμβριος 2012)

Αυτή ήταν μια πολύ μεγάλη αναβάθμιση η οποία μπορούσε να συνδεθεί σε Domains των windows και να έχει πλήρη υποστήριξη. Η τελευταία έκδοση ήταν η 4.1.4 η οποία βγήκε τον Ιανουάριο του 2014

Samba TNG (Τέλη 1999)

Αυτή η έκδοση αναπτύχθηκε μετά από διαφωνίες που υπήρχαν μεταξύ της ομάδας Samba και του Luke Leighton. Σκοπός του ήταν να ξαναγράψει τον κώδικα του Samba Domain, αλλά αυτό ήταν αρκετά δύσκολο γιατί δεν υπήρχαν αρκετοί προγραμματιστές που να το υποστηρίζουν, αλλά και επειδή το Samba βασίζεται σε αντίστροφη μηχανική και έτσι δεν υπήρχε κάποιο σχετικό έγγραφο με το πώς λειτουργεί. Η ανάπτυξή του σταμάτησε στην έκδοση 0.5-rc1 η οποία ήταν τον Δεκέμβριο του 2009.

2.1.2 Πρωτόκολλα

Το Samba μπορεί να μοιράζεται αρχεία και εκτυπωτές ανάμεσα σε υπολογιστές Windows και Unix και περιέχει πολλά διαφορετικά πρωτόκολλα.

NetBios μέσω TCP/IP (NBT)

Το NetBios είναι το μοναδικό όνομα που χρησιμοποιεί ο κάθε υπολογιστής στο δίκτυο για πιο εύκολη αναγνώριση από τα προγράμματα. Οι θύρες επικοινωνίας που χρησιμοποιούνται είναι οι UDP 137, 138 και TCP 137, 139.

Server Message Block (SMB)

Το SMB είναι το πρωτόκολλο το οποίο είναι υπεύθυνο για τον διαμοιρασμό των αρχείων, εκτυπωτών, θυρών και διάφορων τηλεπικοινωνιακών συσκευών ανάμεσα στους συνδεδεμένους στο δίκτυο υπολογιστές. Επίσης αυτό είναι υπεύθυνο για την αυθεντικοποίηση του τερματικού που συνδέονται στον υπολογιστή που γίνεται κοινή χρήση.
Χρησιμοποιεί την θύρα 445 ή μπορεί και να χρησιμοποιήσει τις θύρες του NBT.

Common Internet File System (CIFS)

Το πρωτόκολλο CIFS χρησιμοποιείται για τον ορισμό του τρόπου επικοινωνίας μέσω πακέτων με το SMB πρωτόκολλου.

Microsoft Remote Procedure Call (MSRPC)

Είναι μια τροποποιημένη έκδοση του DCE/RPC (Distributed Computing Environment / Remote Procedure Calls) το οποίο χρησιμεύει στο να στέλνει κλήσεις απομακρυσμένης διαδικασίας ανάμεσα στους υπολογιστές σε ένα το δίκτυο. Η Microsoft έφτιαξε το MSRPC το οποίο το αναβάθμισε για να έχει περισσότερες δυνατότητες.

Windows Internet Name Service (WINS)

Είναι μια υπηρεσία σαν το DNS που βασίζεται στο NetBIOS Name Service και δίνει μοναδικά ονόματα στις διευθύνσεις του τοπικού δικτύου. Υπάρχει ένας κεντρικός υπολογιστής ο οποίος χρησιμοποιώντας το DHCP μπορεί να ενημερώνεται για τις διευθύνσεις IP που έχουν και να ενημερώνει την βάση δεδομένων WINS.

Security Accounts Manager (SAM) database

Είναι ένα αρχείο με αποθηκευμένα σε μορφή Hash τους κωδικούς του κάθε χρήστη. Επειδή ο αλγόριθμος Hash είναι μονόδρομος, γι’ αυτό παρέχει κάποια ασφάλεια στην αποθήκευση των κωδικών. Αυτό το αρχείο είναι κλειδωμένο και κρυπτογραφημένο με κλειδί από το σύστημα καθώς λειτουργεί, ωστόσο υπάρχουν τρόποι ανάκτησης του.

Local Security Authority (LSA) service

Η υπηρεσία LSA είναι υπεύθυνη για την αύξηση της ασφάλειας του συστήματος. Οι λειτουργίες που επιτελεί είναι ο έλεγχος των χρηστών που είναι συνδεδεμένοι, η διαχείριση των αλλαγών των κωδικών και η καταγραφή στατιστικών. Στα windows XP η διεργασία είναι η lsass.exe η οποία άλλαξε σε Credential Provider στα Windows Vista/7/8.

Workgroup

Βασίζεται στην λειτουργία Peer-to-Peer και χρησιμοποιείται κυρίως σε μικρά δίκτυα με στόχο την ανεξαρτησία του κάθε υπολογιστή από τους υπολοίπους. Σε μεγαλύτερα δίκτυα υπάρχει πρόβλημα διαχείρισης.

Active Directory (AD)

Το Active Directory σε αντίθεση με το Workgroup, λειτουργεί σε έναν κεντρικό υπολογιστή στον οποίο συνδέονται τα υπόλοιπα τερματικά. Με χρήση του AD αυξάνει τον αριθμό των τερματικών που μπορούν να εξυπηρετηθούν. Υπάρχει επίσης δυνατότητα διαχείρισης των τερματικών που βρίσκονται στο δίκτυο από τον κεντρικό υπολογιστή.

2.1.3 Τρόπος λειτουργίας διαμοιρασμού μέσω Samba

Ο χρήστης του εκάστοτε κεντρικού υπολογιστή (server) διαλέγει ποιους φακέλους θέλει να μοιράσει και επιλέγει προαιρετικά κάποιο username και password καθώς και τα δικαιώματα που θα έχει ο κάθε χρήστης που μπορεί να τα λάβει (client). Όλα αυτά τα κάνει τροποποιώντας το αρχείο smb.conf.

Όπως παρατηρούμε υπάρχει σχέση server - client μεταξύ των υπολογιστών. Ένας υπολογιστής μπορεί να είναι ταυτόχρονα server καθώς και client σε κάποιον άλλο server μέσα στο δίκτυο.

Η διαδικασία σύνδεσης σε διαμοιραζόμενο φάκελο είναι διαφορετική σε Windows και Linux. Στα Linux πρέπει πρώτα να γίνει mount ο φάκελος και μετά μπορούν να εμφανιστούν τα περιεχόμενά του, ενώ στα Windows τα βλέπει σαν κανονικούς φακέλους συστήματος. Και στις δύο περιπτώσεις εφόσον υπάρχει κωδικός στον φάκελο θα ζητηθεί σε ένα παράθυρο popup.

2.2 GTK+

Είναι μια αντικειμενοστραφής εργαλειοθήκη widget γραμμένη σε C για την δημιουργία γραφικού περιβάλλοντος σε οποιαδήποτε πλατφόρμα (Windows, Linux, Mac OS, HTML5). Το GTK+ καθώς και το QT είναι δύο από τις πιο δημοφιλείς εργαλειοθήκες κυρίως σε display servers με Wayland και X11.

2.2.1 Ιστορία

Αρχικά ονομαζόταν GIMP Toolkit και φτιάχτηκε από τον Spancer Kimball και τον Peter Mattis στο πανεπιστήμιο της Καλιφόρνιας. Δημιουργήθηκε για το πρόγραμμα επεξεργασίας εικόνων GIMP το 1997 για να αντικαταστήσει την εργαλειοθήκη Motif και τα κατάφερε να το αντικαταστήσει εντελώς στην έκδοση 0.60 του GIMP. Στην έκδοση 0.99 ξαναγράφτηκε για να είναι αντικειμενοστραφές και άλλαξε το όνομά του σε GTK+. Στην έκδοση 2.0.0 έφτιαξαν αρκετά προβλήματα που υπήρχαν και το μετέτρεψαν σε UTF-8 για να υποστηρίζει πολλές γλώσσες. Στην έκδοση 3.0.0 υποστήριξαν τα θέματα βασισμένα σε CSS και υποστήριξαν καλύτερα το Wayland Display Server.

2.2.2 Τρόπος Λειτουργίας

Για τον σχεδιασμό του γραφικού περιβάλλοντος υπάρχουν αρκετά προγράμματα όπως το Glade, Gazpacho, Crow Designer, Stetic και άλλα. Όλα αυτά τα εργαλεία δημιουργούν ένα αρχείο xml το οποίο ο builder της κάθε γλώσσας που θέλουμε να χρησιμοποιήσουμε το διαβάζει και το μετατρέπει σε τέτοια μορφή έτσι ώστε να το χρησιμοποιήσουμε.

2.2.3 Που χρησιμοποιείται

Είναι πολύ διαδεδομένο στα Linux και πιο συγκεκριμένα σε Desktop Environments, σε Window Managers, σε εφαρμογές συστήματος για το Gnome, όπως και σε πολλές άλλες εφαρμογές.

2.2.4 Επεξήγηση Ονομασιών [4]

Desktop Environment

Είναι ένα πακέτο προγραμμάτων το οποίο τρέχει πάνω στον ίδιο Window Manager και παρέχει δυνατότητες να ελέγχει το σύστημα, αν και δεν παρέχει πλήρεις δυνατότητες, γι αυτό χρησιμοποιούμε το τερματικό. Για τον σχεδιασμό των εφαρμογών του desktop environment χρησιμοποιούνται εργαλεία που λειτουργούν με widgets όπως το glade χρησιμοποιήθηκε στην παρούσα εργασία. Κάποια χαρακτηριστικά παραδείγματα είναι τα παρακάτω:

- Cinnamon
- Gnome
- LXDE

Window Manager

Είναι ένα πρόγραμμα του συστήματος το οποίο ελέγχει την τοποθεσία και την εμφάνιση των παραθύρων στο γραφικό περιβάλλον. Κάποια χαρακτηριστικά παραδείγματα είναι τα παρακάτω:

- Compiz
- Metacity
- Awesome

Display Server

Είναι η διεπαφή ανάμεσα στον πυρήνα του λειτουργικού και το γραφικό περιβάλλον του χρήστη. Αυτό που κάνει είναι να ελέγχει και να προωθεί την είσοδο και την έξοδο ανάμεσα στον χρήστη και στο υλικό του συστήματος.

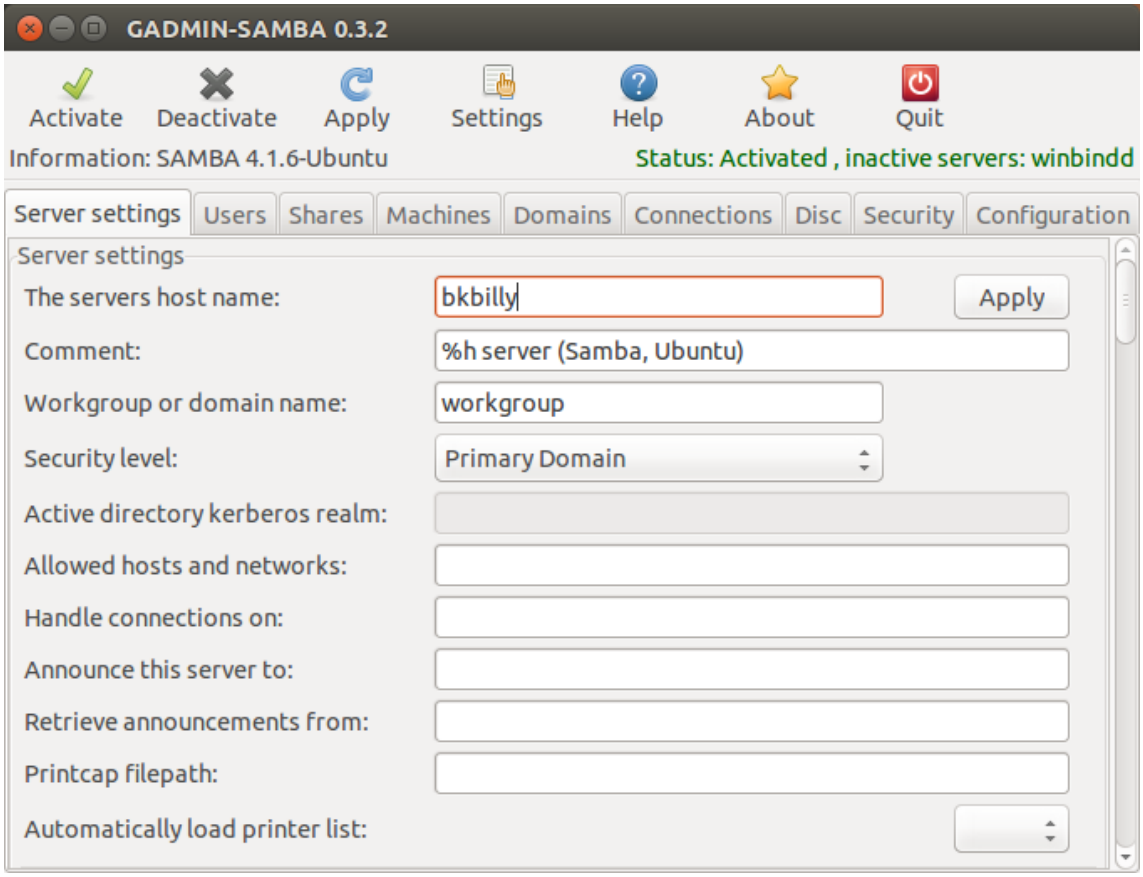
- X11
- Wayland
- Mir
- SurfaceFlinger
- Quartz Compositor

3 Ανάλυση αντίστοιχων εφαρμογών

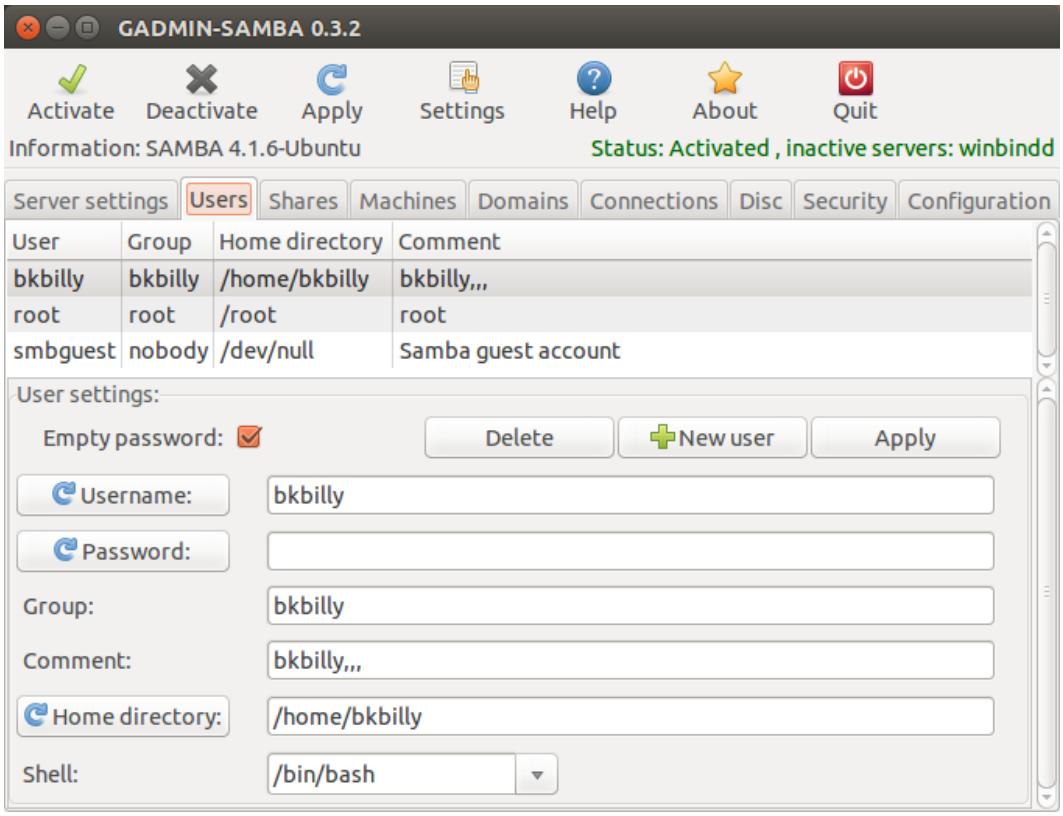
3.1 GADMIN-SAMBA

Αυτή η εφαρμογή είναι μια από τις καλύτερες που υπάρχουν σε θέμα δυνατοτήτων. Έχει ρυθμίσεις για οτιδήποτε θέλουμε να κάνουμε στο samba, αλλά το πρόβλημα είναι οτι είναι αρκετά περίπλοκη στην λειτουργία, μιας και οι περισσότεροι χρήστες δεν ξέρουν τι να κάνουν με όλο αυτό τον όγκο ρυθμίσεων που παρέχει.

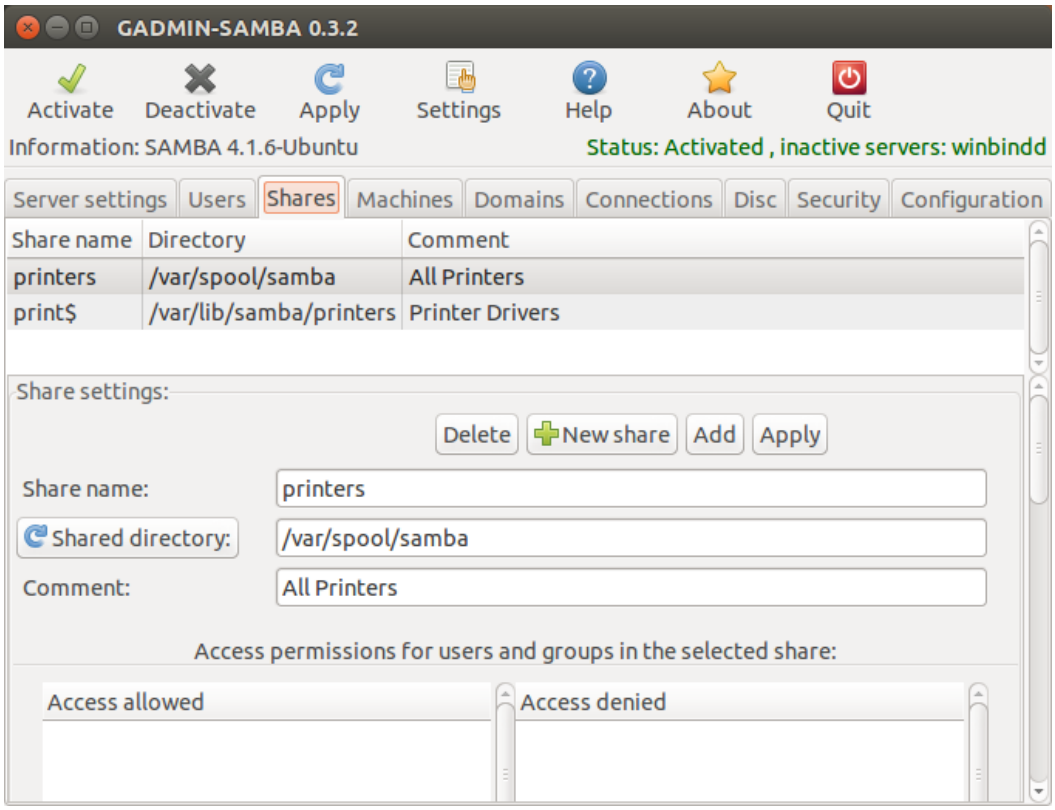
Παρακάτω υπάρχουν κάποιες από τις πιο σημαντικές εικόνες για την λειτουργία της εφαρμογής:



Εικόνα 1: Το κυρίως παράθυρο με τις απαραίτητες ρυθμήσεις



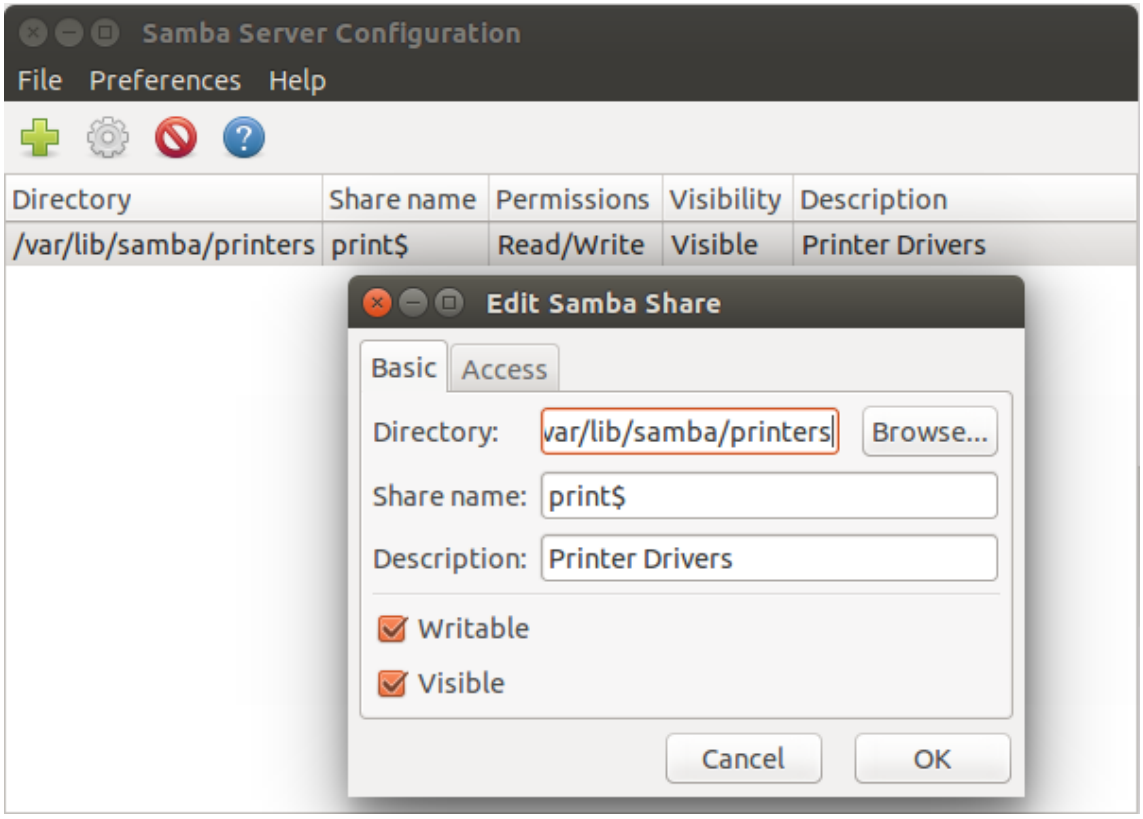
Εικόνα 2: Διαχείριση των διαθέσιμων χρηστών



Εικόνα 3: Διαχείριση των διαμοιραζόμενων φακέλων

3.2 System Config Samba

Αυτή η εφαρμογή είναι πολύ απλή στην χρήση της και γι αυτό είναι και η πιο χρησιμοποιούμενη, μόνο που δεν παρέχει όλες αυτές τις δυνατότητες που παρέχει η GADMIN-SAMBA. Μία πολύ σημαντική δυνατότητα που της λείπει είναι ο διαμοιρασμός εκτυπωτών.



Εικόνα 4: Κυρίως παράθυρο με το παράθυρο διαχείρισης των διαμοιραζόμενων φακέλων

4 Πρώτα βήματα

Ξεκίνησα το πρόγραμμα εγκαθιστώντας το πακέτο quickly [5][6] το οποίο το έφτιαξε η Canonical, η ίδια εταιρία που φτιάχνει το Ubuntu OS. Το quickly είναι μια διεπαφή για ενοποίηση της Python, του GTK και του Glade σε ένα εργαλείο με απλές εντολές.

Οι βασικές εντολές που χρησιμοποιούνται μέσα από το τερματικό για ένα καινούριο project είναι οι ακόλουθες.

- Για να δημιουργήσουμε ένα καινούριο project με όνομα foo απλά γράφουμε:

  ```
  quickly create ubuntu-application foo
  ```

  η οποία δημιουργεί την δομή των αρχείων και βάζει και κάποια χρήσιμα αρχεία όπως τον builder ο οποίος είναι για να μετατρέπει το glade αρχείο σε μορφή που να την διαβάζει η python.
- Εφόσον πάμε στο ίδιο φάκελο με το project που δημιουργήσαμε στο προηγούμενο βήμα, για να σχεδιάσουμε την εφαρμογή μας γράφουμε:

  ```
  quickly design
  ```

  το οποίο ανοίγει το πρόγραμμα glade.
- Για να ανοίξουμε τον κώδικα για να τον αλλάξουμε γράφουμε:

  ```
  qucikly edit
  ```

  το οποίο ανοίγει το αρχείο με τον κώδικα που πρέπει να αλλάξουμε με τον προεπιλεγμένο επεξεργαστή κειμένων.
- Και η τελευταία χρήσιμη εντολή είναι για να τρέξουμε το πρόγραμμα με την εντολή:

  ```
  quickly run
  ```

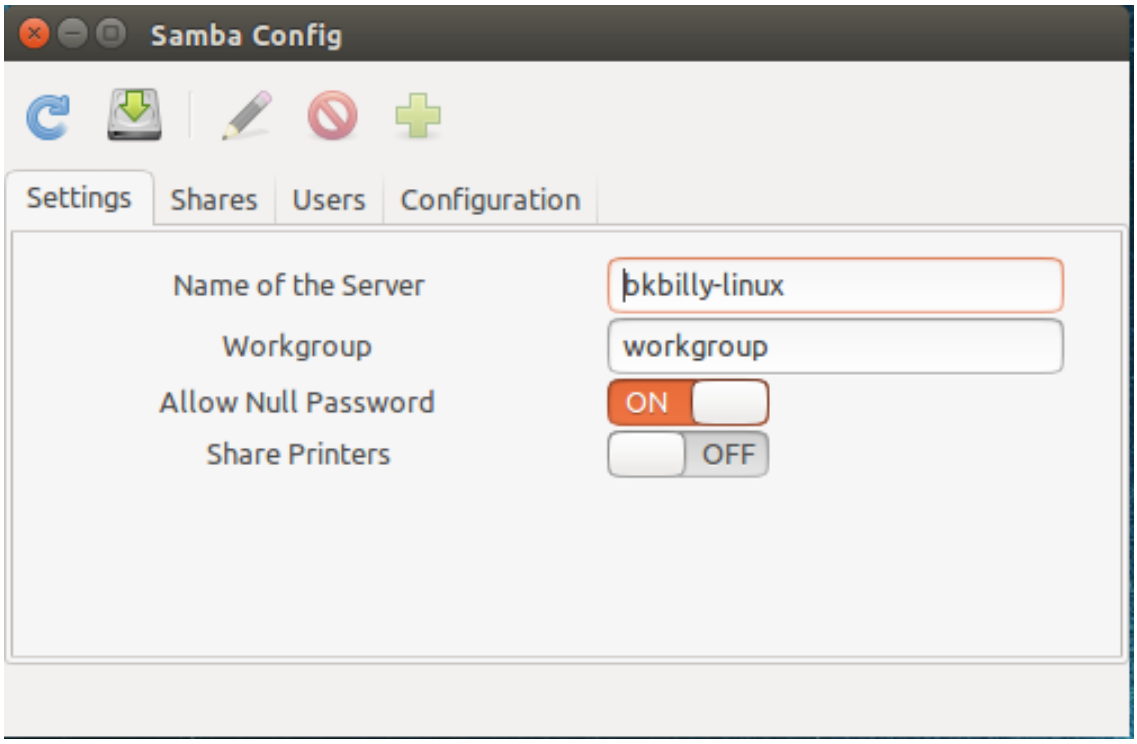
Για να απλοποιήσω και για να καταλάβω τον κώδικα έλεγξα ποια αρχεία χρησιμοποιούνται από το quickly για να λειτουργήσει η εφαρμογή και τα άλλαξα για να τρέχουν με τον δικό μου τρόπο. Έτσι κατέληξα σε δύο αρχεία, το run.py και το builder.py. Το builder έχει τις κατάλληλες μεθόδους για να διαβάζει το αρχείο glade και

να το μετατρέπει σε μορφή που μπορεί να το διαβάσει η python. Το run.py καλεί τον builder και περιέχει όλη την εφαρμογή που έχω γράψει.

4.1 Λειτουργικότητα

4.1.1 Settings Tab

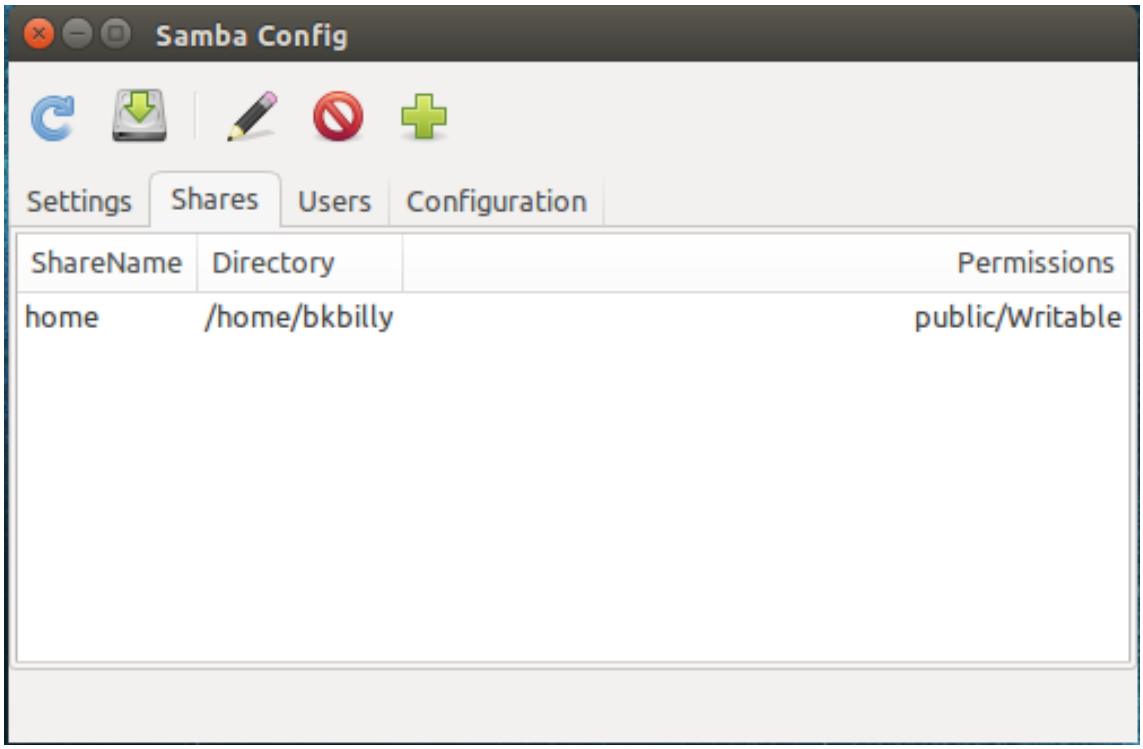
Με το που τρέξουμε την εφαρμογή εμφανίζεται το κυρίως παράθυρο με επιλεγμένη καρτέλα το Settings το οποίο περιέχει ρυθμίσεις για το όνομα του υπολογιστή, σε ποιο workgroup ανήκει, αν επιτρέπεται η σύνδεση με κάποιο φάκελο χωρίς κωδικό και επιλογή για αν θα μοιράζονται οι εκτυπωτές. Στο συγκεκριμένο παράθυρο υπάρχουν στην μπάρα πάνω δύο κουμπιά τα οποία είναι για να επαναφέρουμε τις αλλαγές που έχουμε κάνει ή και να τις σώσουμε, αντίστοιχα.



Εικόνα 5: Το κυρίως παράθυρο με τις απαραίτητες ρυθμίσεις

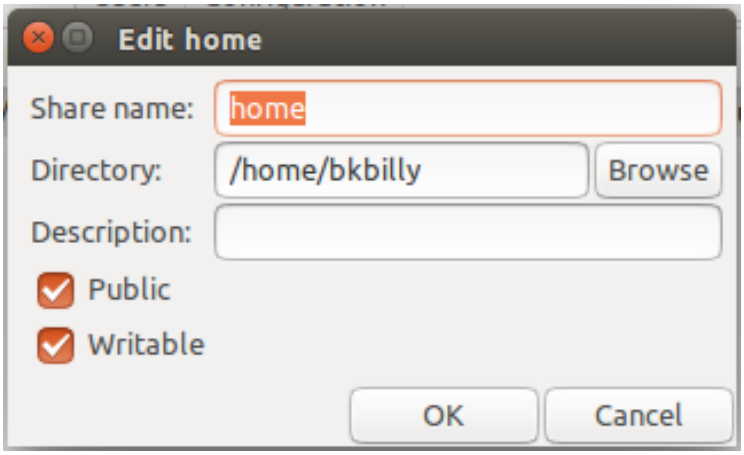
4.1.2 Shares Tab

Στην επόμενη καρτέλα που είναι το Shares επιλέγουμε από τα πάνω κουμπιά στην μπάρα να κάνουμε επεξεργασία του επιλεγμένου φακέλου από την λίστα από κάτω, να τον διαγράψουμε ή και να προσθέσουμε καινούριο.



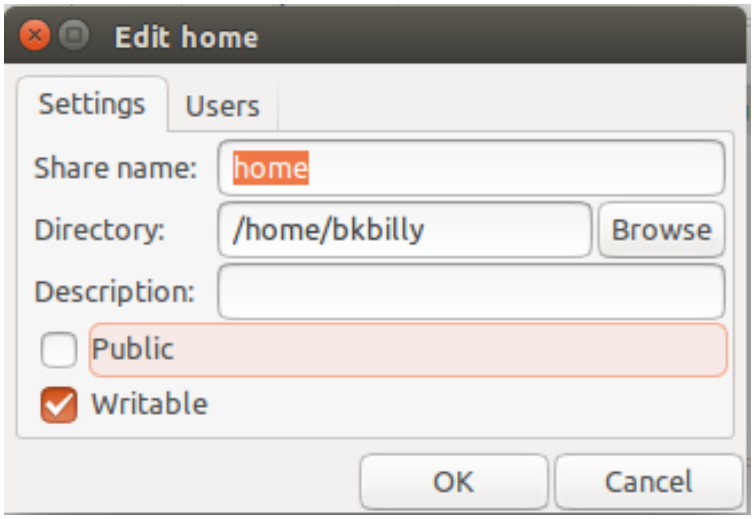
Εικόνα 6: Εμφάνιση διαθέσιμων διαμοιραζόμενων φακέλων με τα δικαιώματά τους

Αν κάνουμε επεξεργασία μπορούμε να αλλάξουμε το όνομα του φακέλου, το path στο οποίο βρίσκεται ή και να βάλουμε και περιγραφή. Επίσης διαλέγουμε και τις ιδιότητες σύνδεσης με τα checkbox πιο κάτω.

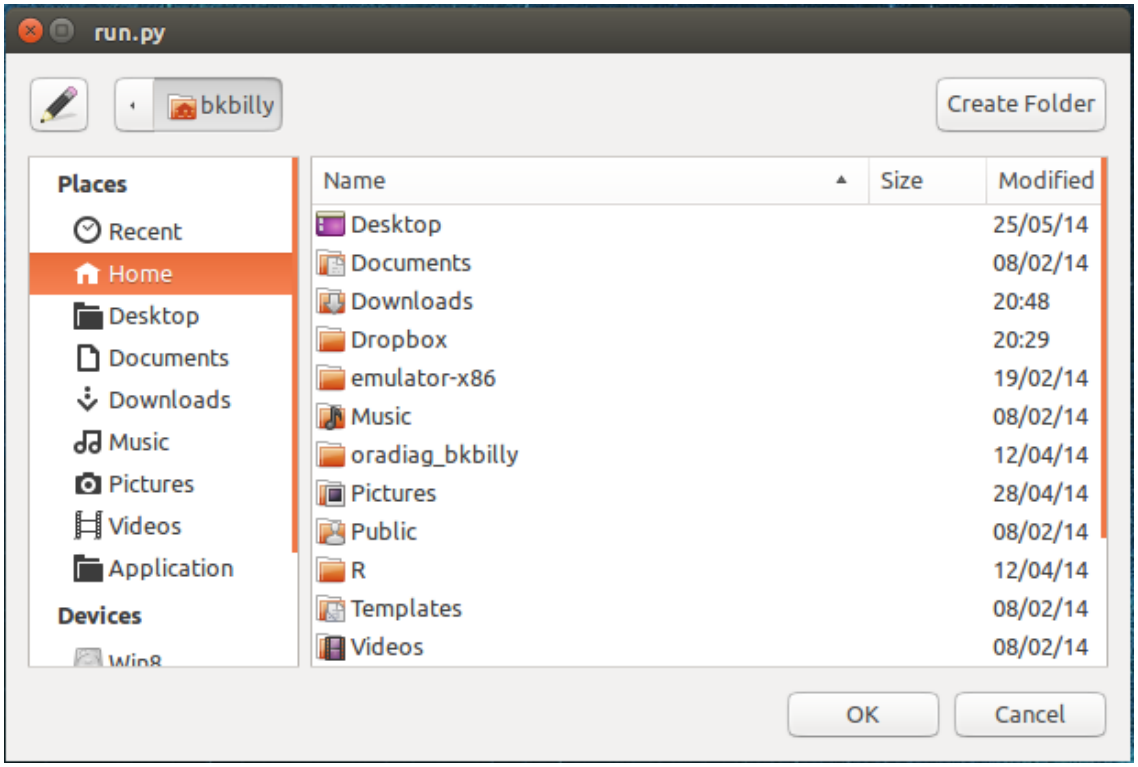


Εικόνα 7: Επεξεργασία ή εισαγωγή κάποιου φακέλου για διαμοιρασμό

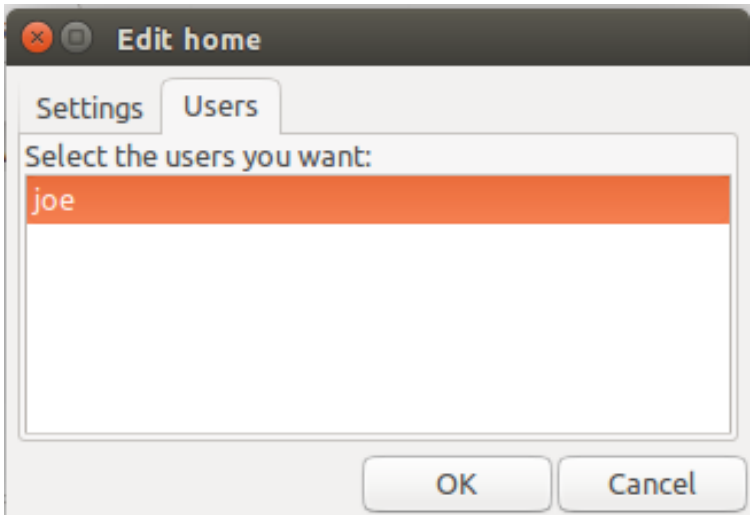
Αν από-επιλέξουμε το Public τότε εμφανίζεται η καρτέλα Users στην οποία θα πρέπει να επιλέξουμε ποιους χρήστες θέλουμε να έχουν πρόσβαση.



Εικόνα 8: Εμφάνιση καρτέλας όταν δεν είναι δημόσιο για επιλογή χρηστών



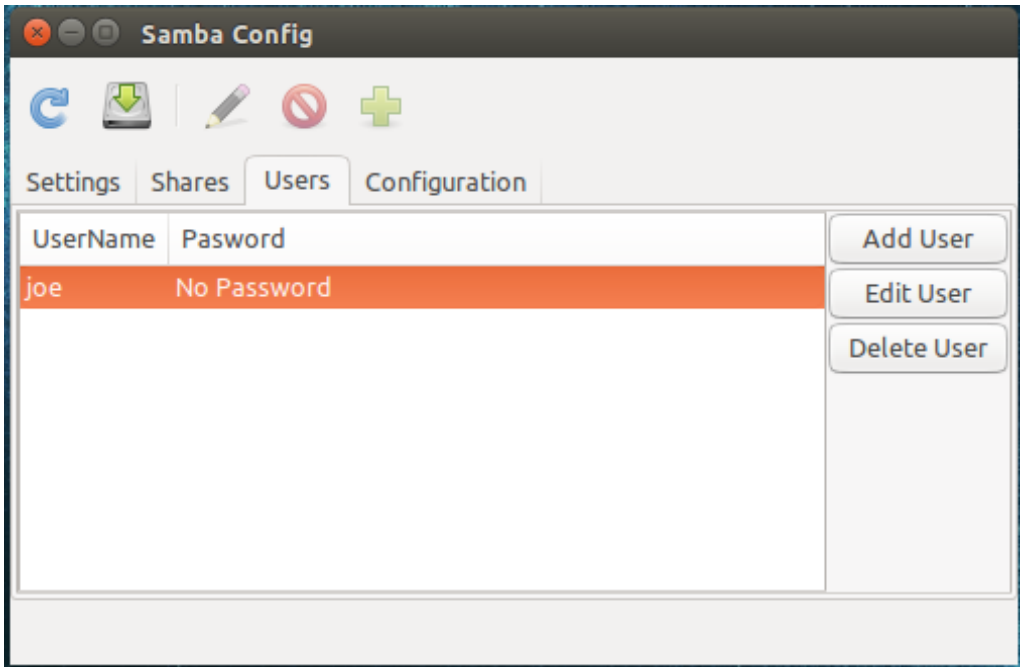
Εικόνα 9: Επιλογή φακέλου για διαμοιρασμό όταν πατηθεί το κουμπί Browse



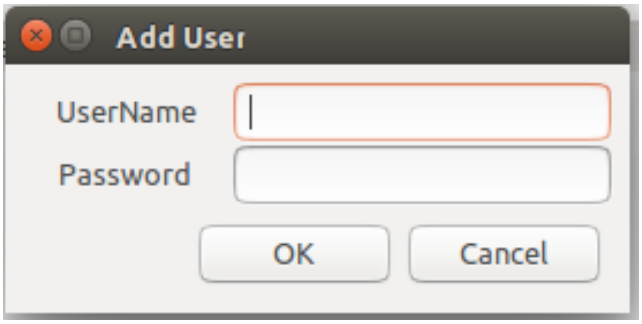
Εικόνα 10: Επιλογή χρηστών

4.1.3 Users Tab

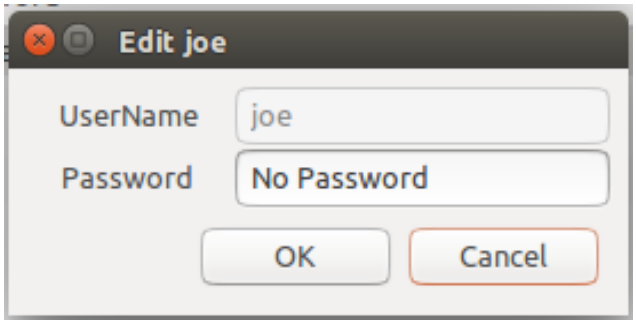
Στην καρτέλα Users μπορούμε να προσθέσουμε κάποιον χρήστη, να τον επεξεργαστούμε ή και να τον διαγράψουμε αφότου τον επιλέξουμε από την λίστα στα αριστερά.



Εικόνα 11: Δημιουργία/Διαγραφή χρηστών και αλλαγή κωδικού



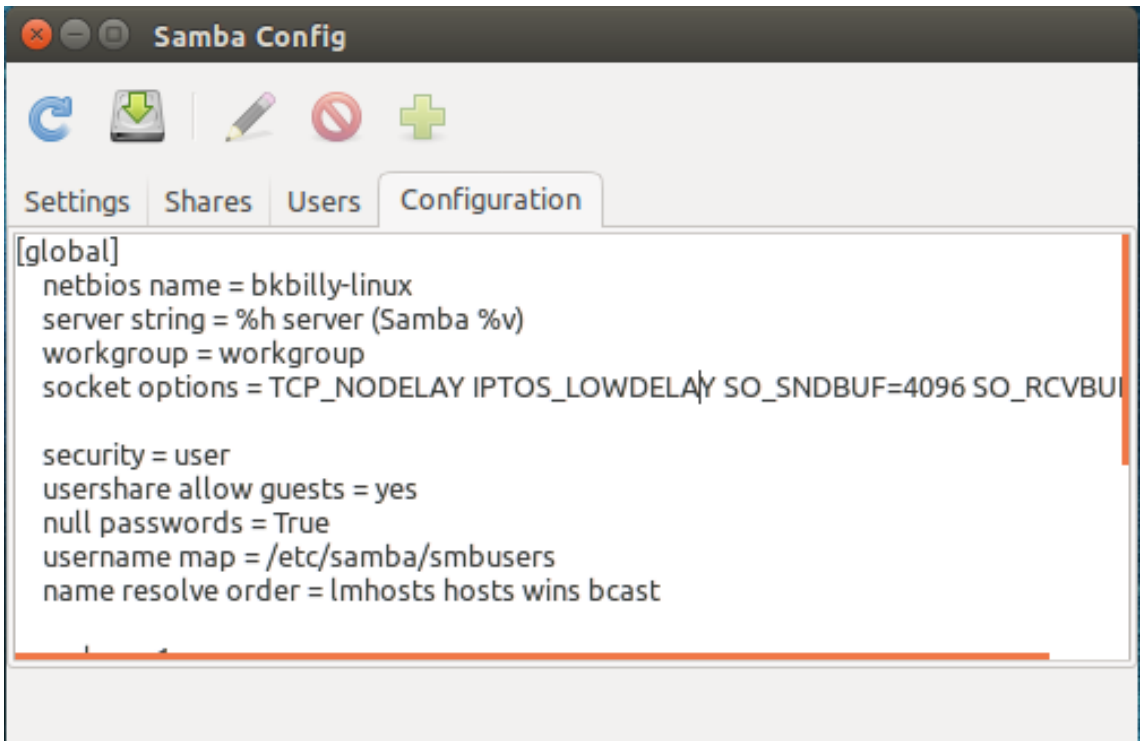
Εικόνα 12: Δημιουργία Χρήστη (προαιρετικό πεδίο ο κωδικός)



Εικόνα 13: Αλλαγή κωδικού χρήστη

4.1.4 Configuration Tab

Στην τελευταία καρτέλα μπορούμε να διαβάσουμε το αρχείο ρυθμίσεων που χρησιμοποιούμε αυτή την στιγμή. Όταν σωθούν οι αλλαγές που έχουμε κάνει πιο πάνω, αυτό θα αλλάξει.



Εικόνα 14: Τελικό αποτέλεσμα που θα αποθηκευτεί στο αρχείο **/etc/samba/smb.conf**

5 Κώδικας

5.1 Εισαγωγή

Οι βιβλιοθήκες που χρειαστήκαμε είναι:

- gi.repository.Gtk [3]
- subprocess.Popen [7]
- subprocess.PIPE [7]
- os [8]
- builder.builder

Υπάρχουν 3 κλάσεις σε όλη την εφαρμογή:

- Window
- WinFunc
- builder

Υπάρχουν 4 παράθυρα σε όλη την εφαρμογή

- samba_config_window
- ShareOptions
- UserOptions
- BrowseDialog

Το πρόγραμμα ξεκινάει καλώντας την κλάση Window και αρχικοποιώντας το Gtk.

```
editor = Window()

Gtk.main()
```

5.2 Class Window

Με το που καλέσουμε την κλάση Window τρέχει οι ακόλουθες εντολές οι οποίες καλούνε τον builder δίνοντας σαν input το αρχείο glade που θα διαβάσει και το όνομα του παραθύρου που θέλουμε να κάνει compile. Στην συνέχεια λέμε στον builder να μας φέρει όλες τις ιδιότητες των widget του glade αρχείου στην τοπική μεταβλητή ui. Μετά αρχικοποιούμε το Password του συστήματος και καλούμε την τελευταία κλάση, την WinFunc στην οποία της δίνουμε τις τοπικές μεταβλητές και το Password.

```
def __init__(self):
   self.MainWindow = builder("SambaConfigWindow.glade","samba_config_window")
   self.ui = self.MainWindow.get_ui(self)


   PASSWORD = "1"
   WinFunc.Initialize(self, PASSWORD)
```

Στην ίδια κλάση υπάρχουν όλα τα events που γίνονται μέσα στο πρόγραμμα και τα οποία είναι τα ακόλουθα:

on_samba_config_window_destroy

Κλείνει το πρόγραμμα εντελώς όταν πατηθεί το X στο παράθυρο

```
Gtk.main_quit()
```

on_mnu_close_activate

Κλείνει το πρόγραμμα εντελώς όταν πατηθεί το close από το menu bar

```
Gtk.main_quit()
```

on_ShareOptions_delete_event

Κρύβει το παράθυρο ShareOptions όταν πατηθεί το X στο παράθυρο

```
self.ui.ShareOptions.hide()
return True
```

on_BrowseDialog_delete_event

Κρύβει το παράθυρο BrowseDialog όταν πατηθεί το X στο παράθυρο

```
self.ui.BrowseDialog.hide()
return True
```

on_UserOptions_delete_event

Κρύβει το παράθυρο UserOptions όταν πατηθεί το X στο παράθυρο

```
self.ui.UserOptions.hide()

return True
```

on_SaveButton_clicked

Καλεί την μέθοδο SaveChanges από την κλάση WinFunc όταν πατηθεί το κουμπί SaveButton από το αρχικό παράθυρο

```
WinFunc.SaveChanges(self)
```

on_RefreshButton_clicked

Καλεί την μέθοδο Initialize από την κλάση WinFunc όταν πατηθεί το κουμπί RefreshButton από το αρχικό παράθυρο

```
WinFunc.Initialize(self, self.ROOTPASS)
```

on_EditButton_clicked

Καλεί την μέθοδο EditShares από την κλάση WinFunc όταν πατηθεί το κουμπί EditButton από το αρχικό παράθυρο

```
WinFunc.EditShares(self)
```

on_DeleteButton_clicked

Καλεί την μέθοδο DeleteShares από την κλάση WinFunc όταν πατηθεί το κουμπί DeleteButton από το αρχικό παράθυρο

```
WinFunc.DeleteShares(self)
```

on_AddButton_clicked

Καλεί την μέθοδο AddShares από την κλάση WinFunc όταν πατηθεί το κουμπί AddButton από το αρχικό παράθυρο

```
WinFunc.AddShares(self)
```

on_SharesTree_focus_out_event

Με το που φύγουμε από το tab των Shares τότε απενεργοποιούμε τα κουμπιά EditButton, DeleteButton και AddButton

```
self.ui.EditButton.set_sensitive(False)

self.ui.DeleteButton.set_sensitive(False)

self.ui.AddButton.set_sensitive(False)
```

on_SharesTree_focus_in_event

Με το που εστιάσουμε στο tab των Shares τότε ενεργοποιούμε τα κουμπιά EditButton, DeleteButton, AddButton και καλούμε και την μέθοδο WriteShares από την κλάση WinFunc

```
self.ui.EditButton.set_sensitive(True)
```

```
self.ui.DeleteButton.set_sensitive(True)
self.ui.AddButton.set_sensitive(True)
WinFunc.WriteShares(self)
```

on_AddUserButton_clicked

Καλεί την μέθοδο AddUser από την κλάση WinFunc όταν πατηθεί το κουμπί AddUserButton από το αρχικό παράθυρο

```
WinFunc.AddUser(self)
```

on_EditUserButton_clicked

Καλεί την μέθοδο EditUser από την κλάση WinFunc όταν πατηθεί το κουμπί EditUserButton από το αρχικό παράθυρο

```
WinFunc.EditUser(self)
```

on_DeleteUserButton_clicked

Καλεί την μέθοδο DeleteUser από την κλάση WinFunc όταν πατηθεί το κουμπί DeleteUserButton από το αρχικό παράθυρο

```
WinFunc.DeleteUser(self)
```

on_SharesCheckPublic_toggled

Με το που πατήσουμε το checkbox Public στο παράθυρο Shares, ελέγχει την κατάσταση στην οποία θα είναι αφού πατηθεί και αν είναι ενεργοποιημένο τότε κρύβει το tab Users, αλλιώς το εμφανίζει.

```
if (self.ui.SharesCheckPublic.get_active()==True):
   self.ui.TabsNotebook.set_show_tabs(False)
else:
   self.ui.TabsNotebook.set_show_tabs(True)
```

on_SharesButtonOK_clicked

Καλεί την μέθοδο ConfirmShareChanges από την κλάση WinFunc όταν πατηθεί το κουμπί SharesButtonOK από το παράθυρο Shares

```
WinFunc.ConfirmShareChanges(self)
```

on_SharesButtonBrowse_clicked

Με το που πατηθεί το κουμπί SharesButtonOK από το παράθυρο Shares, ανοίγει το παράθυρο BrowseDialog με αρχικό φάκελο το directory που είναι γραμμένο στο διπλανό textbox. Αν δεν είναι γραμμένο τίποτα, τότε ανοίγει τον φάκελο “/home”.

```
self.MainWindow.ShowWindow("BrowseDialog")
Directory = self.ui.SharesEntryDirectory.get_text()
```

```
if(Directory == ""):
   self.ui.BrowseDialog.set_current_folder("/home")
else:
   self.ui.BrowseDialog.set_current_folder(Directory)
```

on_SharesButtonCancel_clicked

Κάνει το ίδιο που κάνει το on_ShareOptions_delete_event που κρύβει το παράθυρο.

```
self.ui.ShareOptions.hide()
```

on_BrowseOK_clicked

Με το πατηθεί το κουμπί BrowseOK από το παράθυρο BrowseDialog, ελέγχει αν η διαδρομή του φακέλου υπάρχει και κρύβει το παράθυρο, αλλιώς εμφανίζεται μια ειδοποίηση.

```
try:
   self.ui.SharesEntryDirectory.set_text(self.ui.BrowseDialog.get_current_folder())
   self.ui.BrowseDialog.hide()
except:
   output = Popen(["notify-send", "This is not a Directory"]).communicate()
```

on_BrowseCancel_clicked

Κάνει το ίδιο που κάνει το on_BrowseDialog_delete_event που κρύβει το παράθυρο.

```
self.ui.BrowseDialog.hide()
```

on_UserButtonOK_clicked

Καλεί την μέθοδο ConfirmUserChanges από την κλάση WinFunc όταν πατηθεί το κουμπί UserButtonOK από το παράθυρο Users

```
WinFunc.ConfirmUserChanges(self)
```

on_UserButtonCancel_clicked

Κάνει το ίδιο που κάνει το on_UserOptions_delete_event που κρύβει το παράθυρο.

```
self.ui.UserOptions.hide()
```

5.3 Class WinFunc

Σε αυτή την κλάση γίνονται οι κύριες μέθοδοι του προγράμματος που καλούνται από την Window κλάση. Όποιες δεν καλούνται τις έχω κάνει private. Αυτό στην python γίνεται με δύο κάτω παύλες στην αρχή της κάθε μεθόδου.

Υπάρχουν οι ακόλουθες μέθοδοι σε αυτή την κλάση:

Initialize

Στην αρχή καλείται η κλάση με το password που έχουμε ορίσει στην παραπάνω κλάση και αρχικοποιούμε τις μεταβλητές που θα χρησιμοποιήσουμε. Μετά καλούνται οι απαραίτητες μέθοδοι για να αρχικοποιηθεί η εφαρμογή και να εμφανιστούν στο γραφικό περιβάλλον.

```
self.SMBFILE="/etc/samba/smb.conf"
self.ROOTPASS=ROOTPASS+"\n"
self.NOPASS="No Password"
self.HIDEPASS="*******"
self.SMBLIST=[]
self.SharesList=[]

self.__CreateSMBLIST()
self.__CreateSharesList()
self.__WriteUsers()
self.__WriteConfig()
self.ui.NetbiosName_Entry.set_text(self.__GetProperties("netbios"))
self.ui.Workgroup_Name.set_text(self.__GetProperties("workgroup"))
self.ui.AllowPassword_Switch.set_active(self.__GetProperties("null passwords"))
self.ui.SharePrinters_Switch.set_active(self.__PrintShareExists())
```

__CreateSMBLIST

Σε αυτή την μέθοδο διαβάζει το αρχείο smb.conf του συστήματος και φτιάχνει μία λίστα που περιέχει λίστες. Η κάθε εμφολευμένη λίστα το πρώτο αντικείμενό της είναι το section από το αρχείο smb και τα υπόλοιπα είναι οι παράμετροι με τις τιμές τους. Παρακάτω είναι ένα παράδειγμα της τελικής μορφής της λίστας που φτιάξαμε.

```
[
```

```
   [  '[global]',
      'netbios name = bkbilly-linux',
      'server string = %h server (samba %v)',
      'workgroup = workgroup',
      'socket options = tcp_nodelay iptos_lowdelay so_sndbuf=4096 so_rcvbuf=4096',
      'security = user',
      'usershare allow guests = yes',
      'null passwords = true',
      'username map = /etc/samba/smbusers',
      'name resolve order = lmhosts hosts wins bcast',
      'syslog = 1',
      'syslog only = yes',
      'wins support = yes',
      'usershare owner only = false'
   ],
   [  '[home]',
      'path = /home/bkbilly',
      'public = yes',
      'read only = no'
   ]
]
```

Αφότου τελειώσουμε με την ανάγνωση του αρχείου smb.conf καλούμε την μέθοδο __FixSynonyms.

```
f = open(self.SMBFILE,"r")
for line in f:
   if (line.replace(" ","")!="\n"):
      line=line.strip()
      if (line.find("#")!=0 and line.find(";")!=0):
         if(line[0]=="["):
            templist=[]
            self.SMBLIST.append(templist)
         templist.append(line.lower())
f.close()
self.__FixSynonyms()
```

```
print(self.SMBLIST)
```

__FixSynonyms

Επειδή πολλοί από τις παράμερους έχουν και κάποια συνώνυμα, γι' αυτό το λόγο χρειάστηκε να μετατρέψω σε μια μορφή όλα όσα είχαν συνώνυμα για να μπορώ να τα διαχειριστώ πιο εύκολα. Στην επίσημη σελίδα υπάρχουν οι αντιστοιχήσεις: http://www.samba.org/samba/docs/man/manpages-3/smb.conf.5.html

Οι αλλαγές που έκανα είναι οι ακόλουθες:

```
browseable -> browsable

create mode -> create mask

directory mode -> directory mask

guest ok -> public

directory -> path

print ok -> printable

write ok -> read only

writeable -> read only
```

Και ο κώδικας για να το κάνω είναι ο ακόλουθος:

```
for title in self.SMBLIST:
   for item in title:
      titleID=self.SMBLIST.index(title)
      itemID=title.index(item)
      if(item.find("browseable")==0):
         self.SMBLIST[titleID][itemID] = item.replace("browseable","browsable")
      elif(item.find("create mode")==0):
         self.SMBLIST[titleID][itemID] = item.replace("create mode","create mask")
      elif(item.find("directory mode")==0):
         self.SMBLIST[titleID][itemID] = item.replace("directory mode","directory mask")
      elif(item.find("guest ok")==0):
         self.SMBLIST[titleID][itemID] = item.replace("guest ok","public")
      elif(item.find("directory")==0):
         self.SMBLIST[titleID][itemID] = item.replace("directory","path")
      elif(item.find("print ok")==0):
         self.SMBLIST[titleID][itemID] = item.replace("print ok","printable")
      elif(item.find("write ok")==0):
         self.SMBLIST[titleID][itemID] = item.replace("write ok","read only")
```

```
      elif(item.find("writeable")==0 or item.find("writable")==0):
         invert = item.split('= ')[1]
         if (invert == "no"):
            self.SMBLIST[titleID][itemID] = "read only = yes"
         elif (invert == "yes"):
            self.SMBLIST[titleID][itemID] = "read only = no"
```

__CreateSharesList

Αυτή η μέθοδος φτιάχνει μια λίστα που περιέχει κάθε φάκελο με τις ιδιότητές του που έχουμε κάνει share με την παρακάτω δομή:

```
[['home', '/home/bkbilly', '', 'public', 'Writable']]
```

Και ο κώδικας για να το κάνει αυτό γίνεται αφότου τρέχει πρώτα το CreateSMBLIST και το FixSynonyms που φτιάχνουν το SMBLIST:

```
for title in self.SMBLIST:
   if (title[0].find("[global]")!=0 and title[0].find("[print")!=0):
      ShareName=""
      Directory=""
      Permissions="Writable"
      Comment=""
      Access="User Access"
      ShareName=title[0][1:-1]
      for item in title:
         if(item.find("path")==0):
            Directory=item.split('= ')[1]
         elif(item.find("comment")==0):
            Comment=item.split('= ')[1]
         elif(item.find("public")==0):
            if(item.split('= ')[1]=="yes"):
               Access="public"
         elif(item.find("valid users")==0):
            Access=item.split("= ")[1]
         elif(item.find("read only")==0):
            if(item.split('= ')[1]=="yes"):
               Permissions="Read Only"
      self.SharesList.append([ShareName,Directory,Comment,Access,Permissions])
```

__WriteConfig

Αυτή η μέθοδος διαβάζει το αρχείο ρυθμίσεων του samba όπως είναι και το γράφει στην τελευταία καρτέλα της εφαρμογής.

```
self.ui.ShowConfig_textview.get_buffer().set_text("")
f = open(self.SMBFILE,"r")
for line in f:
   self.ui.ShowConfig_textview.get_buffer().\
   insert(self.ui.ShowConfig_textview.get_buffer().get_end_iter(), line)
f.close()
```

__GetProperties

Αυτή η μέθοδος παίρνει σαν μια μεταβλητή που ψάχνει να την βρει είτε σαν section είτε σαν παράμετρο μέσα στην λίστα που είχαμε φτιάξει πιο πάνω. Αν το βρει, τότε το επιστρέφει, αλλιώς επιστρέφει το μήνυμα Not Found.

```
for title in self.SMBLIST:
   for item in title:
      if(search.find('[')==0):
         if(item==search):
            return item
      elif(item.find(search)==0):
         item = item.split('= ')[1]
         if (item=="true" or item=="yes"):
            return True
         elif (item=="false" or item=="no"):
            return False
         else:
            return item
return "Not Found"
```

__PrintShareExists

Ψάχνει στην λίστα που είχαμε φτιάξει αν υπάρχουν οι ρυθμίσεις για να μοιράζονται τον εκτυπωτή και επιστρέφει True ή False.

```
if (self.__GetProperties("printing")=="cups" and
   self.__GetProperties("printcap name")=="cups" and
   self.__GetProperties("[print$]")== "[print$]" and
```

```
   self.__GetProperties("[printers]") == "[printers]"):
   return True
else:
   return False
```

WriteShares

Διαβάζει την λίστα SharesList και γράφει στην καρτέλα Shares με την κατάλληλη μορφή όλους του φακέλους που μοιράζονται.

```
self.ui.SharesListStore.clear()
for share in self.SharesList:
   self.ui.SharesListStore.append([share[0],share[1],share[3]+"/"+share[4]])
```

DeleteShares

Με αυτή την μέθοδο διαβάζουμε τι έχουμε επιλέξει από την λίστα στην καρτέλα Shares με τους φακέλους και τα διαγράφει και από εκεί, αλλά και από την λίστα SharesList.

```
(model,paths)=self.ui.SharesTree.get_selection().get_selected_rows()
iter = model.get_iter(paths[0])
ShareName = model.get_value(iter,0)

for share in self.SharesList:
   if(share[0]==ShareName):
      break

self.SharesList.remove(share)
model.remove(iter)
```

AddShares

Ανοίγει ένα καινούριο κενό παράθυρο όταν πατηθεί το κουμπί add στο κύριο παράθυρο στην πάνω μπάρα.

```
self.__OpenShares("","","","","","Add Shares")
```

EditShares

Με αυτή την μέθοδο διαβάζουμε τι έχουμε επιλέξει από την λίστα στην καρτέλα Shares με τους φακέλους και ανοίγει ένα καινούριο παράθυρο με τις ιδιότητες που παίρνει από την λίστα SharesList.

```
(model,paths)=self.ui.SharesTree.get_selection().get_selected_rows()
iter = model.get_iter(paths[0])
```

```
ShareName = model.get_value(iter,0)
for share in self.SharesList:
   if(share[0]==ShareName):
      break
self.__OpenShares(share[0],share[1],share[2],share[3],share[4],"Edit "+ShareName)
```

__OpenShares

Παίρνει σαν είσοδο το όνομα του φακέλου (Name), την τοποθεσία του φακέλου (Directory), την περιγραφή του (Description), τους χρήστες που επιτρέπονται για τον συγκεκριμένο φάκελο (Access), τα δικαιώματα που υπάρχουν για τον συγκεκριμένο φάκελο (Permissions) και τον τίτλο του παραθύρου (Title). Στην συνέχεια τα γράφει στα αντίστοιχα πεδία στο παράθυρο.

```
self.ui.TabsNotebook.set_current_page(0)
self.ui.ShareOptions.set_title(Title)
self.ui.SharesEntryName.set_text(Name)
self.ui.SharesEntryDirectory.set_text(Directory)
self.ui.SharesEntryDescription.set_text(Description)
if(Access=="public" or Access==""):
   self.ui.SharesCheckPublic.set_active(True)
else:
   print Access
   self.ui.SharesCheckPublic.set_active(False)
if(Permissions=="Read Only"):
   self.ui.SharesCheckWritable.set_active(False)
else:
   self.ui.SharesCheckWritable.set_active(True)

self.MainWindow.ShowWindow("ShareOptions")
```

ConfirmShareChanges

Καλείται με το που πατηθεί το κουμπί ok στο παράθυρο που προσθέτουμε ή επεξεργαζόμαστε τους φακέλους.

Στην αρχή διαβάζουμε όλες τις ρυθμίσεις από την πρώτη καρτέλα του παραθύρου.

```
Title = self.ui.ShareOptions.get_title()
ShareName = self.ui.SharesEntryName.get_text()
```

```
Directory = self.ui.SharesEntryDirectory.get_text()
Description = self.ui.SharesEntryDescription.get_text()
if(self.ui.SharesCheckPublic.get_active()==True):
   Access="public"
else:
   Access="User Access"
if(self.ui.SharesCheckWritable.get_active()==True):
   Permissions="Writable"
else:
   Permissions="Read Only"
```

Μετά ελέγχουμε αν υπάρχει και δεύτερη καρτέλα και διαβάζουμε κι από εκεί όλους τους χρήστες και τα αποθηκεύουμε στην μεταβλητή Access με κενά μεταξύ των χρηστών.

```
if (Access != "public"):
   Users = ""
   (model,paths)=self.ui.ShareUsers.get_selection().get_selected_rows()
   paths.reverse()
   for rowIter in paths:
      iter = model.get_iter(rowIter)
      Users = Users + " " + model.get_value(iter,0)
   Access = Users
```

Μετά ελέγχουμε αν έχουν γραφτεί όλα τα απαραίτητα κουτιά.

```
if(ShareName=="" or Directory=="" or Access==""):
   print "Missing !!!!"
   return False
for share in self.SharesList:
   if (share[0]==ShareName and Title.split()[0]=="Add"):
      print "Already in List"
      return False
   if not os.path.exists(Directory):
      print "File Path Doesn't Exist"
      return False
```

Και τέλος προσθέτει ή αλλάζει στην λίστα SharesList τον φάκελο που έχουμε φτιάξει και κρύβει το παράθυρο.

```
if(Title.split()[0]=="Add"):
   self.SharesList.append([ShareName,Directory,Description,Access,Permissions])
   print self.SharesList
   self.ui.ShareOptions.hide()
else:
   i=0
   for share in self.SharesList:
      if(share[0]==Title.split()[1]):
         break
      i+=1
   self.SharesList[i]=[ShareName,Directory,Description,Access,Permissions]
   self.ui.ShareOptions.hide()
```

__WriteUsers

Αυτή η μέθοδος γράφει όλους τους χρήστες που υπάρχουν στο σύστημα στην καρτέλα Users του κυρίους παραθύρου της εφαρμογής. Σε αυτό το σημείο αν δεν βρει κάποιο κωδικό, τότε γράφει "No Password", αλλιώς γράφει "*******"

```
self.ui.UsersListStore.clear()
UserName = Popen("sudo -S pdbedit -Lw".split(), stdout=PIPE,
stdin=PIPE).communicate(self.ROOTPASS)[0]
if (len(UserName)>0):
   UserName=UserName.strip().split("\n")
   for item in UserName:
      if(item.find("WARNING")<0):
         if(item.find("NO PASSWORD")>=0):
            UsesPass=self.NOPASS
         else:
            UsesPass=self.HIDEPASS
         UserName=item[:item.find(":")]
         self.ui.UsersListStore.append([UserName,UsesPass])
```

DeleteUser

Αυτή η μέθοδος καλείται όταν πατηθεί το κουμπί Delete Users στην καρτέλα Users του κυρίως παραθύρου. Αυτό που κάνει είναι να ψάχνει ποιος χρήστης έχει επιλεχθεί και στην συνέχει να τον διαγράφει από το σύστημα και στο τέλος και από την λίστα.

```
 (model,paths)=self.ui.UsersTree.get_selection().get_selected_rows()
paths.reverse()
```

```
iter = model.get_iter(paths[0])

User = model.get_value(iter,0)

output = Popen(("sudo -S smbpasswd -x " + User).split(), stdin=PIPE,
stdout=PIPE).communicate(self.ROOTPASS)[0]

output = output.lower()

print (output)

if(output.find("deleted")>=0):

   model.remove(iter)
```

AddUser

Αυτή η μέθοδος καλείται όταν πατηθεί το κουμπί Add Users στην καρτέλα Users του κυρίως παραθύρου. Αυτό που κάνει είναι να ανοίγει ένα παράθυρο με κενό UserName και Password, οπου το UserName το κάνει να μπορεί να γραφτεί.

```
self.ui.UserEntry.set_sensitive(True) #Used to be able to edit the User Input

self.__OpenUsers("","","Add User")
```

EditUser

Αυτή η μέθοδος καλείται όταν πατηθεί το κουμπί Edit Users στην καρτέλα Users του κυρίως παραθύρου. Αυτό που κάνει είναι να ανοίγει ένα παράθυρο με το UserName και Password, όπου το UserName το κάνει να μην μπορεί να γραφτεί.

```
 (model,paths)=self.ui.UsersTree.get_selection().get_selected_rows()

iter = model.get_iter(paths[0])

UserName = model.get_value(iter,0)

Password = model.get_value(iter,1)


self.ui.UserEntry.set_sensitive(False) #Used for not able to edit the UserName Input

self.__OpenUsers(UserName,Password,"Edit "+UserName)
```

__OpenUsers

Καλείται από τις μεθόδους AddUser και EditUser και παίρνει σαν είσοδο το UserName και το Password που τα γράφει στα αντίστοιχα μέρη και από τον τίτλο του παραθύρου (Title).

```
self.ui.UserOptions.set_title(Title)

self.ui.UserEntry.set_text(UserName)

self.ui.PasswordEntry.set_text(Password)

self.MainWindow.ShowWindow("UserOptions")
```

ConfirmUserChanges

Αυτή η μέθοδος καλείται όταν στο παράθυρο που προσθέτει ή αλλάζει τους χρήστες πατηθεί το OK. Αυτό που κάνει είναι να διαβάζει το UserName και το Password που έχει εισαχθεί και μετά ανάλογα με το αν είναι παράθυρο για επεξεργασία ή για προσθήκη χρήστη τρέχει ο αντίστοιχος κώδικας και μετά καλείτε το WriteUsers για να επιβεβαιωθεί η αλλαγή.

```
Title = self.ui.UserOptions.get_title()

UserName = self.ui.UserEntry.get_text()

Password = self.ui.PasswordEntry.get_text()


if (UserName==""): #Error: Do Nothing

   return False

if(Title.split()[0]=="Add"): #Add User

   self.ChangeUser(UserName,Password)

else: #Edit User

   self.ChangeUser(UserName,Password)


self.__WriteUsers()self.ui.UserOptions.hide()
```

ChangeUser

Αυτή η μέθοδος καλείται από την προηγούμενη, την ConfirmUserChanges, η οποία περιέχει όλες τις εντολές για να προσθέσει κάποιον χρήστη ή να αλλάξει το κωδικό του.

Ελέγχει αν ο χρήστης υπάρχει

```
UserExists=False

Users = Popen("sudo -S pdbedit -Lw".split(), stdout=PIPE,
stdin=PIPE).communicate(self.ROOTPASS)[0]

Users=Users.strip().split("\n")

for item in Users:

   Users=item[:item.find(":")].strip()

   if(Users==UserName):

      UserExists=True

      print ("User Already Exists")

      break
```

Αν ο χρήστης υπάρχει, τότε τον προσθέτει στο σύστημα χωρίς κάποιο κωδικό.

```
if(UserExists==False):
```

```
   print "------Add User-------"

   Popen(("sudo -S useradd -r -s /bin/false "+ UserName).split(),
stdin=PIPE).communicate(self.ROOTPASS)

   Popen(("sudo -S smbpasswd -n -a "+ UserName).split(),
stdin=PIPE).communicate(self.ROOTPASS)[0]
```

Ανάλογα με το αν έχει γίνει αλλαγή, αν δεν υπάρχει κωδικός ή αν υπάρχει ο κωδικός τρέχει ο αντίστοιχος κώδικας.

```
if(Password==self.HIDEPASS): #If no change is made to the window's password

   print ("---------Nothing Changed--------")

elif(Password==""): #If no password is requested

   print "------Change UserName-------"

   Popen(("sudo -S smbpasswd -n -a "+ UserName).split(),
stdin=PIPE).communicate(self.ROOTPASS)[0]

   print ("Password Changed to None")

else: #If password is used

   print("-------Changing Password-------")

   smbpasswd = Popen(("sudo -S smbpasswd -s -a "+ UserName).split(), stdin=PIPE)

   smbpasswd.communicate(b'\n'.join([Password, Password]))

   print ("Password Changed to: "+Password)
```

SaveChanges

Αυτή η μέθοδος καλείται όταν πατηθεί το κουμπί save που βρίσκεται στο κυρίως παράθυρο στα πάνω κουμπιά το δεύτερο. Αυτό που κάνει είναι να σώζει ότι αλλαγές έχουν γίνει.

Στην αρχή διαβάζει τις ρυθμίσεις από την πρώτη καρτέλα.

```
NetBios_Name = self.ui.NetbiosName_Entry.get_text()

WorkGroup = self.ui.Workgroup_Name.get_text()

Null_Passwords = self.ui.AllowPassword_Switch.get_active()

Printer_Share = self.ui.SharePrinters_Switch.get_active()
```

Στην συνέχεια διαβάζει από ένα αρχείο που έχουμε σαν πρότυπο και το αλλάζει έτσι ώστε να το γράψει με τον τρόπο που θέλουμε.

```
SaveList=[]

f = open("smb.conf","r")

for line in f:

   if(line.find("----------") >= 0):

      if (Printer_Share==True):
```

```
            line = ""
         else:
            break
      elif(line.find("YOUR_HOSTNAME") > 0):
         line = line.replace("YOUR_HOSTNAME",NetBios_Name)
      elif(line.find("WORKGROUP") > 0):
         line = line.replace("WORKGROUP",WorkGroup)
      elif(line.find("NULL_PASS_TRUE") > 0):
         line = line.replace("NULL_PASS_TRUE",str(Null_Passwords))
      SaveList.append(line)
f.close()
```

Μετά διαβάζει την λίστα SharesList και φτιάχνει μια λίστα την SaveList με την μορφή που να μπορεί να την διαβάζει το samba έτσι ώστε να την γράψουμε στο αρχείο smb.conf.

```
#Write Shares
for share in self.SharesList:
   SaveList.append("\n[" + share[0] + "]")
   SaveList.append("\n    path = "+share[1])
   if (share[2] != ""):
      SaveList.append("\n    comment = "+share[2])
   if (share[3].lower() == "public"):
      SaveList.append("\n    public = yes")
   else:
      SaveList.append("\n    valid users = " + share[3])
   SaveList.append("\n    "+share[4].lower()+" = yes\n")
```

Τέλος διαβάζει την λίστα που φτιάξαμε πιο πριν και την γράφει όπως είναι στο αρχείο /tmp/smb.conf αλλά και στην τελευταία καρτέλα του προγράμματος. Μετά κάνει restart το samba για να πάρει τις καινούριες ρυθμίσεις από το αρχείο.

```
#Save the list to the config file
f = open("/tmp/smb.conf","w")
self.ui.ShowConfig_textview.get_buffer().set_text("")
for item in SaveList:
   f.write(item)
   self.ui.ShowConfig_textview.get_buffer().insert(self.ui.ShowConfig_textview.get_buffe
r().get_end_iter(), item)
```

```
Popen(("sudo -S mv /tmp/smb.conf "+ self.SMBFILE).split(),
stdin=PIPE).communicate(self.ROOTPASS)

Popen(("sudo -S service smbd restart").split(), stdin=PIPE).communicate(self.ROOTPASS)
```

5.4 Class builder

Σε αυτή την κλάση χρησιμοποίησα εκτενώς την συνάρτηση dir() και type() οι οποίες με βοήθησαν στο να καταλάβω τι περιείχε το κάθε αντικείμενο από τον αρχικό builder που έφτιαξε το πακέτο quickly. Αφότου κατάλαβα τι κάνει ακριβώς την πήρα και την άλλαξα για να τρέχει με τον δικό μου τρόπο

Σε αυτή την κλάση υπάρχουν οι ακόλουθες μέθοδοι:

__init__

Αυτή η μέθοδος φτιάχνει ένα αντικείμενο(object) το οποίο περιέχει λεξικό (dict) με το κλειδί (key) του να είναι το όνομα του κάθε widget που βρίσκεται στο παράθυρο που αναλύει (parse). Το κάθε κλειδί περιέχει ένα αντικείμενο με όλες τις ιδιότητες που έχει το κάθε widget.

Η μέθοδος αυτή έχει δύο εισόδους, το όνομα του αρχείου που θέλουμε να αναλύσουμε και το παράθυρο που θέλουμε να ανοίξουμε.

```
self.widgets = {}
self.widgets = {}
self.builder = Gtk.Builder()
self.builder.add_from_file(filename)


self.ShowWindow(window)


tree = ElementTree()
tree.parse(filename)


ele_widgets = tree.getiterator("object")
for ele_widget in ele_widgets:
   name = ele_widget.attrib['id']
   widget = self.builder.get_object(name)


   self.widgets[name] = widget
```

ShowWindow

Αυτή η μέθοδος για να λειτουργήσει πρέπει πρώτα να τρέξει η από πάνω μέθοδος έτσι ώστε να γίνει ανάλυση του αρχείου με τα παράθυρα. Αυτό που κάνει είναι να εμφανίζει το παράθυρο.

```
self.builder.get_object(window).show()
```

get_ui

Αυτό που κάνει είναι να επιστρέφει όλες τις ιδιότητες των widget έτσι ώστε να μπορούμε να τα χρησιμοποιήσουμε στην εφαρμογή μας. Στο τέλος ενώνει την κάθε ιδιότητα του κάθε widget με triggers με μια συγκεκριμένη μορφή.

```
'''Return the attributes of the widgets and Connects to Handler'''
#Get all methods of class window and write them to a dictionary
methods = []
for k in dir(callback_obj):
   try:
      attr = getattr(callback_obj, k)
   except:
      continue
   if inspect.ismethod(attr):
      methods.append((k, attr))
methods.sort()
dict_methods = dict(methods)
callback_handler_dict = {}
callback_handler_dict.update(dict_methods)



for (widget_name, widget) in self.widgets.items():
   setattr(self, widget_name, widget) #Return the attribute
   signal_names = self.__get_signals(widget) #Get a list of the widget signals


   for signal_name in signal_names:
      signal_name = signal_name.replace("-", "_")
      handler_names = ["on_%s_%s" % (widget_name, signal_name)]


      #For every method of the window, connect the signals
```

```
      for handler_name in handler_names:
         target = handler_name in callback_handler_dict.keys()
         if target:
            widget.connect(signal_name, callback_handler_dict[handler_name])

return self
```

__get_signals

Αυτή η μέθοδος επιστρέφει τα signals(triggers) στην παραπάνω μέθοδο.

```
'''Return the signals of the widget'''
signal_ids = []
try:
   widget_type = type(widget)
   while widget_type:
      signal_ids.extend(GObject.signal_list_ids(widget_type))
      widget_type = GObject.type_parent(widget_type)
except RuntimeError:
   pass
signal_names = [GObject.signal_name(sid) for sid in signal_ids]
return signal_names
```

6 Συμπεράσματα

Η ανάπτυξη της εφαρμογής ήταν περισσότερο χρονοβόρα απ' ότι ανέμενα αρχικά, αλλά θεωρώ ότι άξιζε επειδή με βοήθησε να εξελίξω σε σημαντικό βαθμό τις προγραμματιστικές μου δυνατότητες σε python και να κατανοήσω πώς λειτουργούν οι εφαρμογές με γραφικό περιβάλλον, καθώς δεν θα μπορούσα να αποκτήσω τέτοια γνώση με έτοιμα εργαλεία όπως visual studio όπου ο builder του γραφικού περιβάλλοντος γίνεται αυτόματα.

Ένα δύσκολο κομμάτι ήταν ότι δεν υπήρχε κάποιο καλό εγχειρίδιο για το ποιες μεθόδους έχει το κάθε αντικείμενο και έπρεπε κάθε φορά που έβρισκα ένα καινούριο αντικείμενο (BrowseDialog, DeleteButton κλπ) να ψάχνω με την εντολή dir. Επίσης ένα άλλο δύσκολο κομμάτι ήταν και τα triggers για κάθε κίνηση του παραθύρου που έπρεπε να καταλάβω τι ονόματα φτιάχνει. Σε αυτό με βοήθησε το πρόγραμμα glade που έχει όλα τα triggers που χρησιμοποιούνται.

Βιβλιογραφία

[1] http://msdn.microsoft.com/en-us/library/windows/desktop/aa365233(v=vs.85).aspx - Samba

[2] https://wiki.samba.org/index.php/Main_Page - Samba

[3] http://python-gtk-3-tutorial.readthedocs.org/en/latest/index.html - GTK python coding

[4] http://en.wikipedia.org/wiki/Template:X_desktop_environments_and_window_managers - Linux Desktop GUI

[5] http://blog.didrocks.fr/post/Build-your-application-quickly-with-Quickly:-Inside-Quickly-part-1 - Ubuntu quickly

[6] https://wiki.ubuntu.com/Quickly - Ubuntu quickly

[7] https://docs.python.org/2/library/subprocess.html - python subprocess

[8] https://docs.python.org/2/library/os.html - python os library