LDAP Documentation ©2003

University Visualization and Animation Group
Virginia Tech Backsburg, Virginia
by
Patrick Shinpaugh

Overview

LDAP (lightweight directory access protocol) was used to provide directory services and allow common access to home directories as well as a common login for all machines in the cluster. The use of NFS (network file sharing) is also required to export directories to the cluster LDAP clients.

All documentation assumes the use of Redhat. If another distro or UNIX is being used the paths to commands may be different, or the distro may have another method to start/stop/restart processes/daemons or the entire solution may vary.

Linux Installation/Configuration

For the current configuration RedHat Linux 8.0 was installed on the machines. Custom installation was chosen with X and KDE installed. All development options were chosen. In addition to the custom installation, glut, glut-devel, and openldap-clients were installed on the LDAP client machines. For the LDAP server and openldap-servers were also installed in addition to the client dependencies. The LDAP Configuration Utility requires Gtk-Perl gtkglarea be installed.

LDAP Server

> **Important Note** - Because the home directories will be stored in the /export/home directories rather than /home, if a separate partition is to be set up for the home directories be sure to set up the partition as /export/home instead.
>
> **Important Note** - To log into an LDAP client requires that the LDAP server be running slapd (ldap daemon). Otherwise, access to the client machine will be impossible even by the root superuser. To gain access to the machine it will be

necessary to reboot into single user mode and then modify the /etc/pam.d/system-auth file. Simply comment out each line containing pam_ldap.so. When the LDAP server is up and running again, uncomment each of the lines containing pam_ldap.so and restart autofs (automount daemon).

Important Note - Because of issues with the use of ldap in conjunction with a firewall, the firewall should be disabled in all runlevels. Run redhat-config-services, stop the iptables service and remove the checkmark nect to iptables for all runlevels. This issue can be resolved but a good solution has not yet been implemented.

Important Note – The root superuser account does not have superuser abilities when accessing the directories exported by LDAP to client machines. Only on the LDAP server does the root superuser have superuser abilities with respect to the exported directories.

There are several packages necessary for LDAP to be installed on the server. These are nss_ldap, openldap-servers, openldap-clients, authconfig, autofs, and nfs. To run the LDAP configuration utility (Appendix C) written by Luke Schaarf and updated by Patrick Shinpaugh, you will also need to install Gtk-Perl and gtkglarea.

Before modifying the systems to use LDAP, you should copy all existing home directories from /home to /export/home as the /home directories will disappear when LDAP is started resulting in the inability to access that data. The data does remain on the drive but will only be accessible when autofs and ldap are stopped. You may also want to back up the data to some form of removeable media.

The LDAP server was also set up as a client so that users could log into the server/console machine. The /etc/openldap/slapd.conf (Appendix A) and /etc/openldap/ldap.conf (Appendix A) files need to be modified so that slapd (ldap daemon) will run properly for your configuration. The rootpw line of the slapd.conf file should be modified to reflect the password you choose for the Manager account. Use the /usr/sbin/slappasswd command to create an encrypted password to be used as the rootpw in the slapd.conf file.

```
/usr/sbin/slappasswd -h {CRYPT}
```

It will prompt for a password to be entered twice and then display the resulting password hash. You should use the appropriate parameters to retrieve a password formatted correctly for the type of encryption being used. The resulting hash should be copied into the slapd.conf file as the rootpw parameter.

The domain used here at the Virginia Tech visualization lab is sv.vt.edu where a host named thishost would have a hostname of thishost.sv.vt.edu. As such it is only necessary to modify the portion of the files where dc=sv,dc=vt,dc=edu to match your own domain and to change the hostname or ip address of the server to match your own.

A certification key should be created for use with LDAP. A default key exists in the /usr/share/ssl/certs/ directory. It could be used but it is probably better to create a new key if security is a concern. Change the directory to /usr/share/ssl/certs/ and run the command

```
make slapd.pem
```

which will ask you questions which will become a part of the certification key. Answer the questions and the key will be created. You should verify that the owner of the slapd.pem file is ldap and the group is ldap so that the ldap daemon will be able to access the files without a problem. Use the chown command to change the owner/group to ldap:ldap.

After the slapd.conf and ldap.conf files have been modified and the slapd.pem file has been created, slapd can be run from /usr/sbin/ with debug options set if desired to allow you to see useful information should there be problems during the configuration process.

```
/usr/sbin/slapd -d -1
```

The -d -1 argument sets debugging to full. Although this will generate exceptionally large amounts of output, it can be very useful if problems arise during the initial phases of LDAP configuration. As long as it runs and does not error out you can move on to the next step. It is necessary to set up the information that will be tracked through LDAP. It is possible to set up customized schemas but it is unnecessary and much simpler to use the standard existing schemas. The ldapdb.ldif (Appendix A) file is used to set up the database used by LDAP to track information. To set up the database using this file call

```
ldapadd -x -D “dc=sv,dc=vt,dc=edu” -W -f ldapdb.ldif
```

If you ran slapd with the debugging options you should see several lines stating that additions were made. There is a useful tool called LDAP Browser/Editor which can be found at http://www-unix.mcs.anl.gov/~gawor/ldap/. It will allow you to view or edit information stored in the LDAP database. If there were no errors you can shutdown the slapd process either by hitting Ctrl-C where it is running or using the kill command to kill the process. If you look in the /var/lib/ldap directory several .gdbm files will have been created. These are the database files and contain the database information. You should verify that the owner of each .gdbm file is ldap and the group is ldap so that the ldap daemon will be able to access the files without a problem. As the root superuser, use the chown command to change the owner/group to ldap:ldap.

You should modify the /etc/exports (Appendix A) file to specify which directories will be exported. You should use the services control panel (redhat-config-services) to set these three services (ldap, nfs, and autofs) to run when the server is booted. If the server will be running in multiuser text mode (run level 3) you should modify the services to run in both run level 3 and run level 5. It is a good idea to reboot the server and verify that everything will work properly. Otherwise, when you switch run levels some of these services may stop resulting in the inability for users to login to LDAP client machines.

If users are unable to login you will probably only need to restart the automount daemon as well as the ldap and nfs daemons. If that does not solve the problem, check each of the files depending on whether it is a client or server. If the root superuser is unable to login, you should boot into single user mode. Once in single user mode, edit the /etc/pam.d/system-auth file commenting out each of the lines containing pam_ldap.so. This will allow the root superuser the ability to login under run level 3 or 5 and edit whichever files may be causing the LDAP problems, though it should be possible to resolve the problem through single user mode.

The /etc/ldap.conf server file (Appendix A) should be used on the server only if it is also a client. However, it is different than the client file of the same name. If the LDAP server is also to be a client then all of the client files should be modified appropriately except for the /etc/ldap.conf file which should be taken from Appendix A instead of Appendix B.

If the server is also a client then the /etc/ldap.conf (Appendix A), /etc/nsswitch.conf (Appendix B), /etc/auto.master (Appendix B), /etc/auto.home_ldap.pl (Appendix B), and /etc/pam.d/system-auth (Appendix B) files should be modified or added as shown in the Appendices. These files exist in the /home/shpatric/dads-cluster/ldap/server directory on hammer.

LDAP Client

The client machines should be running autofs (automount daemon). The following /etc files should be modified to allow access to the LDAP server. These are /etc/ldap.conf (Appendix B), /etc/nsswitch.conf (Appendix B), /etc/auto.master (Appendix B), and /etc/pam.d/system-auth (Appendix B) files. These files exist in the /home/shpatric/dads-cluster/ldap/server directory on hammer. Because of issues with automount, a Perl script written by Luke Schaarf named auto.home_ldap.pl should be copied into the /etc directory.

You should add the IP address and hostname of the LDAP server to the /etc/hosts file. At this point it is only necessary to restart the network and autofs processes with the following commands

/etc/rc.d/init.d/network restart
/etc/rc.d/init.d/autofs restart

This will restart the automount daemon and update the network protocols and should allow access to the LDAP server.

You should open a new shell and test to see if LDAP is accessible. Attempt to login to the machine as an LDAP user. See the LDAP Configuration Utility section below to learn how to add a user. If the correct hostname and password is supplied and it refuses your connection then there is a problem. If a problem exists, verify that the LDAP server is running slapd. Verify that the hostname and password supplied are correct by logging into the LDAP server with that identity.

LDAP Configuration Utility

The LDAP configuration utility (Appendix C) can be used to add, modify, and delete hosts, groups and users. The slapd process must be running on the LDAP server in order to use the utility program.

To add a host, run the LDAP configuration utility and click on the hosts tab. At the bottom click on the Add button and a dialog box will pop up. You should enter only 2 hosts, the localhost and LDAP server hostname. Enter the hostname in the first text box. The IP address should be entered in the second text box. Enter the hostname again after the cn= in the third text box, but do not erase or alter the remainder of the line. To verify that the host was added properly click on the Refresh button at the bottom of the page.

Add a group by running the LDAP configuration utility and click on the group tab. At the bottom click the Add button and a dialog will pop up. Enter the name of the group in the first text box which could be named user. Enter a number representing the group id of the group which could be 100. And enter the group name again after the cn= in the third text box, but do not erase or alter the remainder of the line. To verify that the group was added properly click on the Refresh button at the bottom of the page.

Finally, add some users. The first dialog box asks for the PID of the user to add. The utility is set up to use the university finger server to find information about an existing individual. If the user being added has a PID and they have not set their information to be confidential then all of their known information will be set automatically. Otherwise, enter the desired username and press the OK button. The next dialog box asks for the userid of the person to add. You should use a

number greater than 1000 and unique to the added user. The last dialog box shows many text boxes. If the finger server was able to find information about the user being added, then most of it will already be filled in. If not, the sn is the last name and is mandatory. To set a password, the Set button should be pressed. It will request that the password be entered twice to confirm your choice of password. There are other mandatory fields...

Appendices

Appendix A. LDAP Server Files

/etc/openldap/slapd.conf

Notes:

To set up the LDAP server in a different domain you should modify the following lines.

```
suffix          "dc=sv,dc=vt,dc=edu"
suffix          "o=CAVE,c=US"
rootdn          "cn=Manager,dc=sv,dc=vt,dc=edu"
rootpw {CRYPT}vOl52ywEvIGjg
```

For the two suffix lines and the rootdn it is necessary to change to match your domain. The rootpw line should be altered to match the encrypted password to be used by the LDAP Manager user. This can be created using the slappasswd command.

```
# $OpenLDAP: pkg/ldap/servers/slapd/slapd.conf,v 1.8.8.7 2001/09/27 20:00:31 kurt Exp $
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#

# Schema
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/redhat/rfc822-MailMember.schema
include         /etc/openldap/schema/redhat/autofs.schema
include         /etc/openldap/schema/redhat/kerberosobject.schema
include         /etc/openldap/schema/samba.schema

# Encryption
TLSCertificateFile /usr/share/ssl/certs/slapd.pem
TLSCertificateKeyFile /usr/share/ssl/certs/slapd.pem
```

```
# Access Control
#defaultaccess auth
#access to attr=userPassword
#       by self write
#       by dn="cn=Manager,dc=sv,dc=vt,dc=edu" write
#       by domain=".*\.sv\.vt\.edu" read
#       by anonymous auth
#       by * none
#access to *
#       by self write
#       by dn="cn=Manager,dc=sv,dc=vt,dc=edu" write
#       by * read


#######################################################################
# ldbm database definitions
#######################################################################

#loglevel       3583
database        ldbm
readonly        off
#password-hash          {CRYPT}
suffix          "dc=sv,dc=vt,dc=edu"
suffix          "o=CAVE,c=US"
rootdn          "cn=Manager,dc=sv,dc=vt,dc=edu"
rootpw          {CRYPT}vOl52ywEvIGjg
disallow        bind_anon
directory       /var/lib/ldap
index   objectClass,uid,uidNumber,gidNumber,memberUid eq
index   cn,mail,surname,givenname                       eq,subinitial
```

/etc/openldap/ldap.conf

Notes:

To set up the LDAP server in a different domain you should modify the following lines.

BASE dc=sv,dc=vt,dc=edu

This line represents a domain of sv.vt.edu where a machine named thishost would have an address of thishost.sv.vt.edu. Simply modify the dc values to correspond to each subaddress within your domain.

```
# $OpenLDAP: pkg/ldap/libraries/libldap/ldap.conf,v 1.4.8.6 2000/09/05 17:54:38 kurt Exp $
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

#BASE           dc=example, dc=com
#URI    ldap://ldap.example.com ldap://ldap-master.example.com:666

#SIZELIMIT  12
#TIMELIMIT 15
#DEREF                  never
HOST 127.0.0.1
BASE dc=sv,dc=vt,dc=edu
```

ldapdb.ldif

Notes:

To set up the LDAP server in a different domain you should modify all of the lines containging dc=sv, dc=vt, dc=edu to match your own domain. Also the userPassword for the Workstation user should be altered by replacing it with the output generated by the slappasswd command used to generate encrypted passwords.

```
dn: dc=sv,dc=vt,dc=edu
dc: dc=sv
objectClass: dcObject
objectClass: organization
o: CAVE

dn: cn=Manager, dc=sv,dc=vt,dc=edu
objectClass: organizationalRole
cn: Manager

dn: cn=Workstation, dc=sv,dc=vt,dc=edu
sn: Workstation
userPassword:: e1NIQX1sSEdaSUFCZDk1aUdLbklGdnpnbGNpbVpONkk9
loginShell: /bin/false
gidNumber: 65536
uidNumber: 65536
objectClass: top
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: organizationalRole
uid: Workstation
gecos: Workstation
cn: Workstation
homeDirectory: /

dn: ou=Aliases, dc=sv,dc=vt,dc=edu
ou: Aliases
objectClass: top
objectClass: organizationalUnit
```

```
dn: ou=Group, dc=sv,dc=vt,dc=edu
ou: Group
objectClass: top
objectClass: organizationalUnit

dn: ou=Hosts, dc=sv,dc=vt,dc=edu
ou: Hosts
objectClass: top
objectClass: organizationalUnit

dn: ou=Mounts, dc=sv,dc=vt,dc=edu
ou: Mounts
objectClass: top
objectClass: organizationalUnit

dn: ou=People, dc=sv,dc=vt,dc=edu
ou: People
objectClass: top
objectClass: organizationalUnit

dn: ou=automount, dc=sv,dc=vt,dc=edu
ou: automount
objectClass: top
objectClass: organizationalUnit
```

/etc/exports

Notes:

To set up the LDAP server in a different domain you should modify all of the lines containging .sv.vt.edu to match your own domain. Also the IP address for the /export/home line should be modified to match the IP address of your LDAP server.

/var/spool/mail *.sv.vt.edu(rw)
/export/home *.sv.vt.edu(rw) 128.173.49.114(rw)
/export *.sv.vt.edu(rw)

/etc/ldap.conf

Notes:

This files should only be used on the LDAP server if it is also going to be a client. To set up the LDAP server in a different domain you should modify the base line dc values to match the dc values for your domain.

```
# @(#)$Id: ldap.conf,v 2.28 2001/08/28 12:17:29 lukeh Exp $
#
# This is the configuration file for the LDAP nameservice
# switch library and the LDAP PAM module.
#
# PADL Software
# http://www.padl.com
#

# Your LDAP server. Must be resolvable without using LDAP.
host 127.0.0.1

# The distinguished name of the search base.
base dc=sv,dc=vt,dc=edu

# Another way to specify your LDAP server is to provide an
# uri with the server name. This allows to use
# Unix Domain Sockets to connect to a local LDAP Server.
#uri ldap://127.0.0.1/
#uri ldaps://127.0.0.1/
#uri ldapi://%2fvar%2frun%2fldapi_sock/
# Note: %2f encodes the '/' used as directory separator

# The LDAP version to use (defaults to 3
# if supported by client library)
#ldap_version 3

# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
#binddn cn=proxyuser,dc=example,dc=com
#binddn cn=Workstation,ou=people,dc=sv,dc=vt,dc=edu
binddn cn=Workstation,dc=sv,dc=vt,dc=edu
```

```
# The credentials to bind with.
# Optional: default is no credential.
#bindpw secret
bindpw 0try,do

# The distinguished name to bind to the server with
# if the effective user ID is root. Password is
# stored in /etc/ldap.secret (mode 600)
#rootbinddn cn=manager,dc=example,dc=com

# The port.
# Optional: default is 389.
#port 389

# The search scope.
#scope sub
#scope one
#scope base

# Search timelimit
#timelimit 30

# Bind timelimit
#bind_timelimit 30

# Idle timelimit; client will close connections
# (nss_ldap only) if the server has not been contacted
# for the number of seconds specified below.
#idle_timelimit 3600

# Filter to AND with uid=%s
#pam_filter objectclass=account

# The user ID attribute (defaults to uid)
#pam_login_attribute uid

# Search the root DSE for the password policy (works
# with Netscape Directory Server)
#pam_lookup_policy yes
```

```
# Group to enforce membership of
#pam_groupdn cn=PAM,ou=Groups,dc=example,dc=com

# Group member attribute
#pam_member_attribute uniquemember

# Template login attribute, default template user
# (can be overriden by value of former attribute
# in user's entry)
#pam_login_attribute userPrincipalName
#pam_template_login_attribute uid
#pam_template_login nobody

# HEADS UP: the pam_crypt, pam_nds_passwd,
# and pam_ad_passwd options are no
# longer supported.

# Do not hash the password at all; presume
# the directory server will do it, if
# necessary. This is the default.
#pam_password clear

# Hash password locally; required for University of
# Michigan LDAP server, and works with Netscape
# Directory Server if you're using the UNIX-Crypt
# hash mechanism and not using the NT Synchronization
# service.
#pam_password crypt

# Remove old password first, then update in
# cleartext. Necessary for use with Novell
# Directory Services (NDS)
#pam_password nds

# Update Active Directory password, by
# creating Unicode password and updating
# unicodePwd attribute.
#pam_password ad

# Use the OpenLDAP password change
# extended operation to update the password.
```

```
#pam_password exop

# RFC2307bis naming contexts
# Syntax:
# nss_base_XXX                  base?scope?filter
# where scope is {base,one,sub}
# and filter is a filter to be &'d with the
# default filter.
# You can omit the suffix eg:
# nss_base_passwd     ou=People,
# to append the default base DN but this
# may incur a small performance impact.
#nss_base_passwd     ou=People,dc=example,dc=com?one
#nss_base_shadow     ou=People,dc=example,dc=com?one
#nss_base_group              ou=Group,dc=example,dc=com?one
#nss_base_hosts              ou=Hosts,dc=example,dc=com?one
#nss_base_services    ou=Services,dc=example,dc=com?one
#nss_base_networks   ou=Networks,dc=example,dc=com?one
#nss_base_protocols  ou=Protocols,dc=example,dc=com?one
#nss_base_rpc        ou=Rpc,dc=example,dc=com?one
#nss_base_ethers     ou=Ethers,dc=example,dc=com?one
#nss_base_netmasks   ou=Networks,dc=example,dc=com?ne
#nss_base_bootparams         ou=Ethers,dc=example,dc=com?one
#nss_base_aliases    ou=Aliases,dc=example,dc=com?one
#nss_base_netgroup   ou=Netgroup,dc=example,dc=com?one

# attribute/objectclass mapping
# Syntax:
#nss_map_attribute    rfc2307attribute        mapped_attribute
#nss_map_objectclass rfc2307objectclass     mapped_objectclass

# configure --enable-nds is no longer supported.
# For NDS now do:
#nss_map_attribute uniqueMember member

# configure --enable-mssfu-schema is no longer supported.
# For MSSFU now do:
#nss_map_objectclass posixAccount User
#nss_map_attribute uid msSFUName
#nss_map_attribute uniqueMember posixMember
#nss_map_attribute userPassword msSFUPassword
```

```
#nss_map_attribute homeDirectory msSFUHomeDirectory
#nss_map_objectclass posixGroup Group
#nss_map_attribute cn msSFUName
#pam_login_attribute msSFUName
#pam_filter objectclass=User
#pam_password ad

# configure --enable-authpassword is no longer supported
# For authPassword support, now do:
#nss_map_attribute userPassword authPassword
#pam_password nds

# For IBM AIX SecureWay support, do:
#nss_map_objectclass posixAccount aixAccount
#nss_base_passwd ou=aixaccount,?one
#nss_map_attribute uid userName
#nss_map_attribute gidNumber gid
#nss_map_attribute uidNumber uid
#nss_map_attribute userPassword passwordChar
#nss_map_objectclass posixGroup aixAccessGroup
#nss_base_group ou=aixgroup,?one
#nss_map_attribute cn groupName
#nss_map_attribute uniqueMember member
#pam_login_attribute userName
#pam_filter objectclass=aixAccount
#pam_password clear

# Netscape SDK LDAPS
#ssl on

# Netscape SDK SSL options
#sslpath /etc/ssl/certs/cert7.db

# OpenLDAP SSL mechanism
# start_tls mechanism uses the normal LDAP port, LDAPS typically 636
#ssl start_tls
#ssl on

# OpenLDAP SSL options
# Require and verify server certificate (yes/no)
# Default is "no"
```

```
#tls_checkpeer yes

# CA certificates for server certificate verification
# At least one of these are required if tls_checkpeer is "yes"
#tls_cacertfile /etc/ssl/ca.cert
#tls_cacertdir /etc/ssl/certs

# SSL cipher suite
# See man ciphers for syntax
#tls_ciphers TLSv1

# Client sertificate and key
# Use these, if your server requires client authentication.
#tls_cert
#tls_key
ssl no
pam_password md5
```

Appendix B. LDAP Client Files

/etc/ldap.conf

Notes:

To set up the LDAP client in a different domain you should modify the base and binddn lines dc values to match the dc values for your domain and change the hostname to match the hostname of your LDAP server.

```
#ssl start_tls
ssl no
pam_password md5
host narf
base dc=sv,dc=vt,dc=edu
binddn cn=Workstation,dc=sv,dc=vt,dc=edu
bindpw 0try,do
```

/etc/nsswitch.conf

Notes:

To set up the LDAP client in a different domain you should modify the base and binddn lines dc values to match the dc values for your domain and change the hostname to match the hostname of your LDAP server.

```
#
# /etc/nsswitch.conf
#
# An example Name Service Switch config file. This file should be
# sorted with the most-used services at the beginning.
#
# The entry '[NOTFOUND=return]' means that the search for an
# entry should stop if the search in the previous entry turned
# up nothing. Note that if the search failed due to some other reason
# (like no NIS server responding) then the search continues with the
# next entry.
#
# Legal entries are:
#
#       nisplus or nis+         Use NIS+ (NIS version 3)
#       nis or yp               Use NIS (NIS version 2), also called YP
#       dns                     Use DNS (Domain Name Service)
#       files                   Use the local files
#       db                      Use the local database (.db) files
#       compat                  Use NIS on compat mode
#       hesiod                  Use Hesiod for user lookups
#       [NOTFOUND=return]       Stop searching if not found so far
#

# To use db, put the "db" in front of "files" for entries you want to be
# looked up first in the databases
#
# Example:
#passwd:    db files nisplus nis
#shadow:    db files nisplus nis
#group:     db files nisplus nis
```

```
passwd:     files ldap
shadow:     files ldap
group:      files ldap

#hosts:     db files nisplus nis dns
hosts:      files dns

# Example - obey only what nisplus tells us...
#services:   nisplus [NOTFOUND=return] files
#networks:   nisplus [NOTFOUND=return] files
#protocols:  nisplus [NOTFOUND=return] files
#rpc:        nisplus [NOTFOUND=return] files
#ethers:     nisplus [NOTFOUND=return] files
#netmasks:   nisplus [NOTFOUND=return] files

bootparams: nisplus [NOTFOUND=return] files

ethers:     files
netmasks:   files
networks:   files
protocols:  files ldap
rpc:        files
services:   files ldap

netgroup:   files ldap

#publickey:  nisplus

automount:  files ldap
aliases:    files
```

/etc/auto.master

Notes:

To set up the LDAP client in a different domain you should modify the base and binddn lines dc values to match the dc values for your domain and change the hostname to match the hostname of your LDAP server.

```
# $Id: auto.master,v 1.2 1997/10/06 21:52:03 hpa Exp $
# Sample auto.master file
# Format of this file:
# mountpoint map options
# For details of the format look at autofs(8).
#/misc /etc/auto.misc --timeout=60

/home program:/etc/auto.home_ldap.pl
```

/etc/auto.home_ldap.pl

Notes:

To set up the LDAP client in a different domain you should modify the base and binddn lines dc values to match the dc values for your domain and change the hostname to match the hostname of your LDAP server.

```
#!/usr/bin/perl -w
#
# Name: auto.home_ldap
#
# Purpose: To retrieve automount information from an LDAP server that refuses
# anonymous binding.  The existing Linux automount client (as of RedHat 7.3)
# can only retrieve automount information if anonymous binding is allowed.
# Since I don't want to hand out user information to just anyone, a password is
# required to connect to the server.
#
# Author: Luke Scharf (luke@vt.edu) May, 2002
#
# License: GNU General Public License (http://www.gnu.org)
#

##### Includes #####
use strict;


##### Constants #####
my $scope  = "subtree";
my $fields = "automountInformation";


##### Variables #####
my $host;
my $binddn;
my $bindpw;
my $base;
my $ssl;
my $username;
my $query;
```

```
##### Get Parameters #####
# Parse command line
my $ARGC = @ARGV;
if ($ARGC < 1) { exit 0; } # Abort if no commands were given
$username = $ARGV[0]; # What user's home directory are we looking for?

# Read system-wide ldap configuration file(s)
open(CONFIG, "< /etc/ldap.conf");
while(<CONFIG>) {
        # Tokenize the line
        @_ = split();

        # Do not process blank or bogus lines
        if ( not $_[0] ) { next; }

        # Store parameters from the config file
        if ( $_[0] =~ /^host$/ ) { $host = $_[1];  }
        if ( $_[0] =~ /^binddn$/ ) { $binddn= $_[1]; }
        if ( $_[0] =~ /^bindpw$/ ) { $bindpw= $_[1]; }
        if ( $_[0] =~ /^base$/ ) { $base= $_[1]; }
        if ( $_[0] =~ /^ssl$/ ) { $ssl = $_[1]; }
}


##### Communicate With LDAP Server #####
# Compose the query
$query = "(&(objectClass=automount)(cn=$username))";

# Glob together a command line with all of the search parameters
my $ldapcommand;
$ldapcommand = "ldapsearch -LLL";
$ldapcommand .= " -h $host";
$ldapcommand .= " -D $binddn";
if ($ssl=~/no/) {
        $ldapcommand .= " -x";
}
$ldapcommand .=      " -w \"$bindpw\"";
$ldapcommand .=      " -b $base";
$ldapcommand .=      " -s $scope";
```

```
$ldapcommand .=       " \'$query\'";
$ldapcommand .=       " $fields";

# Open a pipe to ldapsearch
open(LDAP, "$ldapcommand |");

# Output in an automounter-friendly manner
my $user;
my $home;
while( my $line = <LDAP> ) {
        chomp $line;

        if ($line =~ /^automountInformation\:/) {
                $home = $line;
                $home =~ s|^automountInformation\: ||g;
                print "$home\n";
        }
}
```

/etc/pam.d/system-auth

Notes:

Should there be difficulty logging into an LDAP client as the root superuser, verify that slapd (ldap daemon) is running on the LDAP server machine. If it is then it may be necessary to reboot the machine into single user mode. Edit the /etc/pam.d/system-auth file and comment out the lines containing pam_ldap.so to gain access in multiuser mode. However, this should not be necessary as

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      /lib/security/pam_env.so
auth        sufficient    /lib/security/pam_unix.so likeauth nullok
auth        sufficient    /lib/security/pam_ldap.so use_first_pass
auth        required      /lib/security/pam_deny.so

account     required      /lib/security/pam_unix.so
account     [default=bad success=ok user_unknown=ignore service_err=ignore
system_err=ignore] /lib/security/pam_ldap.so

password    required      /lib/security/pam_cracklib.so retry=3 type=
password    sufficient    /lib/security/pam_unix.so nullok use_authtok md5 shadow
password    sufficient    /lib/security/pam_ldap.so use_authtok
password    required      /lib/security/pam_deny.so

session     required      /lib/security/pam_limits.so
session     required      /lib/security/pam_unix.so
session     optional      /lib/security/pam_ldap.so
```

Appendix C. LDAP Configuration Utility

lldap.pl

Notes:

If LDAP is being set up on a different server then the following line needs to be changed

$self->{DEFAULT_AUTOMOUNTINFORMATION} = "hammer:/export/home/&";

so that hammer is replaced by the name of the server's hostname.

If LDAP is being set up outside of the vt.edu domain then the finger server should be changed to reflect th name of the finger server within the new domain.

$self->{FINGER_SERVER} = "finger.vt.edu";

If a finger server does not exist then the finger server should be set to null (""). It may be necessary to remove or comment out the code which accesses the finger server.

```
#!/usr/bin/perl -w
#
# Name: lldap.pl
#
# Purpose: Easily add users to an LDAP database, as per the standards of the
# Virginia Tech CAVE
#
# Author: Luke Scharf (luke@vt.edu), Feb 2001
# Modified: Patrick Shinpaugh, Dec 2003
#
# License: GPL ( see http://www.gnu.org/copyleft/gpl.html )
#

############### Pre ###############
# Directives
use Gtk;
use strict;


############### Constants ###############
```

```
my $false = 0;
my $true = 1;


############## Main ##############

## Build GUI ##
init Gtk;
my $configuration = new configuration();
my $mainwindow = new tabbed_window(\$configuration);

## Run GUI ##
Gtk::main Gtk;
exit(0);




############## Objects ##############

#################################################################################
# Name: configuration
#
# Purpose: hold configuration parameters for the application
#
{ package configuration;
 sub new{
        my $self = {};

        ##### Search Parameters #####
        $self->{BINDDN} = "cn=Workstation,dc=sv,dc=vt,dc=edu";
        $self->{BINDPW} = "0try,do";
        $self->{SEARCHDN} = "dc=sv,dc=vt,dc=edu";
        $self->{MANAGER_DN} = "cn=Manager,dc=sv,dc=vt,dc=edu";

        ##### LDAP Database layout information ####
        $self->{OU_PEOPLE} = "People";
        $self->{OU_GROUP} = "Group";
        $self->{OU_HOSTS} = "Hosts";
        $self->{PASSWORD_FIELD} = "userPassword";
```

```
        ##### New Account defaults #####
        $self->{DEFAULT_SHELL} = "/bin/bash";
        $self->{DEFAULT_HOME} = "/home/&";
        $self->{DEFAULT_USEAUTOMOUNT} = $true;
        $self->{DEFAULT_AUTOMOUNTINFORMATION} = "hammer:/export/home/&";
        $self->{DEFAULT_GIDNUMBER} = "100";


        ##### Finger Information #####
        $self->{FINGER_SERVER} = "finger.vt.edu";
        $self->{FINGER_UIDFIELD} = "vtAuthNID";


        ##### Programs #####
        $self->{LDAPADD} = "/usr/bin/ldapadd";
        $self->{LDAPSEARCH} = "/usr/bin/ldapsearch";
        $self->{LDAPMODIFY} = "/usr/bin/ldapmodify";
        $self->{LDAPDELETE} = "/usr/bin/ldapdelete";
        $self->{FINGER} = "/usr/bin/finger";

        bless($self);
  }
}



##############################################################################
# Name: tabbed_window
#
# Purpose: Build the main window.
#
# Role: GUI component
#
{ package tabbed_window;

  ############################################################################
  # Name: new
  #
  # Role: constructor
  #
  sub new {
```

```
	# Parameters
	my $self = {};
	bless ($self);
	my ($configuration) = @_;
	bless(\$configuration);

	# Construct Window
	my $window = new Gtk::Window( 'toplevel' );
	$window->set_title( "Luke's LDAP Editor" );
	$window->set_default_size( 800, 600 );
	$window->set_modal($true);
	$window->signal_connect( "delete_event", sub { Gtk->exit( 0 ); } );

	# Construct tab-container
	my $notebook = new Gtk::Notebook();
	$window->add($notebook);
	$notebook->show();

	# Put a pages in the notebook
	$self->makepage($window, $notebook, "People", "posixAccount",
		"uid", "uidNumber", "gidNumber", "sn", "gecos", "title",
		"mail", "loginShell", "homeDirectory", "userPassword");
	$self->makepage($window, $notebook, "automount", "automount",
		"automountInformation");
	$self->makepage($window, $notebook, "Group", "posixGroup", "gidNumber");
	$self->makepage($window, $notebook, "Hosts", "ipHost", "ipHostNumber");


	# Put it on the screen
	$window->show();

	# Cleanup
	return $self;
}



###############################################################################
# Name: makepage
#
# Arguments:
```

```
#       $self - provided by Perl
#       $notebook - the notebook to which we'll add this pae
#       $ou - organizational unit of the current tab
#       $objectClass - object class that we'll search for
#       @columns - the columns we'll pull from the LDAP database
#
# Purpose: Build a page/tab in the interface.
#
# Role: helper for constructor
#
sub makepage {
        # Parameters
        my ($self, $parentwindow, $notebook, $ou, $objectclass, @columns) = @_;

        # Include mandatory items
        unshift @columns, "cn";
        unshift @columns, "dn";

        # vbox (container)
        my $vbox = new Gtk::VBox($false, 0);
        $vbox->show();

        # Scrolled Window
        my $scrolledwindow = new Gtk::ScrolledWindow();
        $vbox->pack_start($scrolledwindow, $true, $true, 0);
        $scrolledwindow->show();

        # columned list
        my $clist = new Gtk::CList($#columns+1);
        $scrolledwindow->add($clist);
        $clist->column_titles_show();
        my $col = 0;
        foreach my $coltext (@columns) {
                $clist->set_column_title($col, $coltext);
                $clist->set_column_auto_resize($col, $true);
                $col++;
        }
        $clist->show();

        # buttonbox (container)
        my $buttonbox = new Gtk::HButtonBox();
```

```
$buttonbox->set_layout_default( 'start' );
$vbox->pack_start($buttonbox, $false, $false, 0);
$buttonbox->show();

# "Refresh" button
my $refreshbutton = new Gtk::Button("Refresh");
$buttonbox->pack_start($refreshbutton, $false, $false, 0);
$refreshbutton->signal_connect( 'clicked', \&refreshbutton_clicked,
                                $ou, $objectclass, $clist, @columns);
$refreshbutton->show();

# "Add" button
my $addbutton = new Gtk::Button("Add");
$buttonbox->pack_start($addbutton, $false, $false, 0);
$addbutton->signal_connect( 'clicked', \&addbutton_clicked,
                $parentwindow, $ou, @columns);
$addbutton->show();

# "Edit" button
my $editbutton = new Gtk::Button("Edit");
$buttonbox->pack_start($editbutton, $false, $false, 0);
$editbutton->signal_connect( 'clicked', \&editbutton_clicked,
                $parentwindow, $ou, $objectclass, $clist, @columns);
$editbutton->show();

# "Delete" button
my $deletebutton = new Gtk::Button("Delete");
$buttonbox->pack_start($deletebutton, $false, $false, 0);
$deletebutton->signal_connect( 'clicked', \&deletebutton_clicked,
                $parentwindow, $ou, $objectclass, $clist);
$deletebutton->show();

# "quit" button
my $quitbutton = new Gtk::Button("Quit");
$buttonbox->pack_start($quitbutton, $false, $false, 0);
$quitbutton->signal_connect( "clicked", sub { Gtk->exit( 0 ); } );
$quitbutton->show();


# Add label & page into notebook
my $label = new Gtk::Label($ou);
```

```
	$notebook->append_page($vbox, $label);

	# Fill
	refreshbutton_clicked($refreshbutton, $ou, $objectclass, $clist, @columns);

	# Return
	return $vbox;
}


################################################################################
# Name: refresh
#
# Arguments:
#	$self - provided by Perl
#	$ou - organizational unit of the current tab
#	$objectClass - object class that we'll search for
#	$clist - the clist that we'll dump the stuff into
#	@cols - the columns we'll pull from the LDAP database
#
# Purpose: Read the LDAP database and dump the output into the provided
#	clist.
#
# Role: Callback
#
sub refreshbutton_clicked{
	# Variables
	my %thisrow;
	my $row;
	my $binddn = $configuration->{BINDDN};
	my $bindpw = $configuration->{BINDPW};
	my $searchdn = $configuration->{SEARCHDN};
	my $ldapsearch = $configuration->{LDAPSEARCH};

	# Parameters
	my ($self, $ou, $objectClass, $clist, @cols) = @_;

	# Clear clist
	$clist->clear();

	# Talk to ldapsearch
```

```
	my $command = "$ldapsearch \\
			-LLL -D \'$binddn\' -x -w \'$bindpw\' \\
			-b \'$searchdn\' \'(objectClass=$objectClass)\'";
	#print "$command\n";
	open( LDAPSEARCH, "$command |");
	while ( <LDAPSEARCH> ){
		# Clean line
		chomp($_);

		# What kind of line is it?
		if ($_ =~ /^dn:/){
			# A start-of-record?
			if ($thisrow{'dn'}){ refresh_addrecord($clist, \%thisrow, \@cols); }

			# bookkeeping
			$thisrow{'dn'} = refresh_datafromline($_);
			$row++;
		}
		elsif (my $col = refresh_iscolumn($_, @cols)){
			# is it a normal row?  Throw it into this hash.
			my $data = refresh_datafromline($_);
			$thisrow{$col} = $data;
		}
	}
	if ($thisrow{'dn'}){ refresh_addrecord($clist, \%thisrow, \@cols); }
	close( LDAPSEARCH );
}
sub refresh_addrecord{
	# Parameters
	my ($clist, $thisrow, $cols) = @_;

	# Add a row
	$clist->append($$thisrow{'dn'});

	# Write the text into the columns
	for my $i (0..$clist->columns-1){
		my $title = $clist->get_column_title($i);
		if (not $$thisrow{$title}){
			$$thisrow{$title} = "";
		}
		$clist->set_text($clist->rows()-1, $i, $$thisrow{$title});
```

```
                $$thisrow{$title} = "";
        }
}
sub refresh_iscolumn{
        # Parameters
        my ($targetline, @cols) = @_;

        # Search
     foreach my $target (@cols){
                if ($targetline =~ /^$target:/){
                        return $target;
                }
        }
        return $false;
}
sub refresh_datafromline{
        # Parameters
        my ($line) = @_;

        # Strip
        chomp($line);
        $line =~ s|^.*: ||g;

        # Return
        return $line;
}

#################################################################################
sub addbutton_clicked{
        # Parameters
        my ($self, $parentwindow, $ou, @fields) = @_; # Parameters

        # Variables
        my $username;
        my %fr;
        my %ed;

        # Create placeholders
        foreach my $i (@fields){
                $ed{$i}=$i;
        }
```

```
$ed{'dn'} = "cn=,ou=$ou,$configuration->{SEARCHDN}";


# are we adding a user account?
if ($ou eq $configuration->{OU_PEOPLE})
{
        ### Finger ###
        # Prompt user for uid
        if (not main::input_box("UID of the user?", \$username,
                        $true, $parentwindow)) {
                return;
        }
        chomp($username);

        # Lookup
        print "username=$username\n";
        %fr = main::finger( $configuration->{FINGER_SERVER}, $username);

        ### Ask User for Real Name ###
        if (not $fr{'FullName'}){
                main::input_box("Full name of user",
                        \$fr{'FullName'},
                        $true, $parentwindow);
        }

        ### Translate from Finger to LDAP ###
        $ed{'dn'} = "uid=$username,ou=$ou,$configuration->{SEARCHDN}";
        #$ed{'gecos'} = $fr{'FullName'};
        #$ed{'cn'} = $fr{'FullName'};
        #$ed{'sn'} = lastname($fr{'FullName'});
        $ed{'gecos'} = $fr{'AlsoKnownAs'};
        $ed{'cn'} = $fr{'AlsoKnownAs'};
        $ed{'sn'} = lastname($fr{'AlsoKnownAs'});
        $ed{'uid'} = $username;
        $ed{'title'} = $fr{'title'};
        $ed{'mail'} = $fr{'EMailAddress'};
        $ed{'uidNumber'} = $fr{$configuration->{FINGER_UIDFIELD}};
        $ed{'gidNumber'} = $configuration->{DEFAULT_GIDNUMBER};
        $ed{'homeDirectory'} = homefix(
                        $configuration->{DEFAULT_HOME},
                        $username);
```

```
                $ed{'loginShell'} = $configuration->{DEFAULT_SHELL};
                $ed{'telephoneNumber'} = $fr{'WorkPhone'};
                $ed{'automountInformation'}
                        = $configuration->{DEFAULT_AUTOMOUNTINFORMATION};
        }


        ### Start edit-window ###
        my $editor = edit_window::new($configuration, $parentwindow, $false, $ou, %ed);
}
sub lastname{
        my ($fullname) = @_;
     my @splitname = split(" ", $fullname);
     my $surname = pop(@splitname);
        return $surname;
}
sub homefix{
        my ($home, $uid) = @_;
        $home =~ s|\&|$uid|g;
        return $home;
}


################################################################################
sub editbutton_clicked{
        # Parameters
        my ($self, $parentwindow, $ou, $objectClass, $clist, @fields) = @_;

        # Variables
        my %ed;

        # Get the current row
        my $row = $clist->selection();
        my $dn = $clist->get_text($row, 0);

        my $col = 0;
        foreach my $field (@fields)
        {
                my $text = $clist->get_text($row, $col);
                $ed{$field} = $text;
                $col++;
```

```
        }

        foreach my $field (keys %ed){
                print "\$ed{$field}: $ed{$field}\n";
        }

        # Use edit dialog
        my $editor = edit_window::new($configuration, $parentwindow, $true, $ou, %ed);
 }

 ###############################################################################
 sub deletebutton_clicked{
        # Parameters
        my ($self, $parentwindow, $ou, $objectClass, $clist) = @_;

        # Get the current row
        my $row = $clist->selection();
        my $dn = $clist->get_text($row, 0);

        # Complain
        main::error_box("FIXME: Delete function not yet implimented.", $parentwindow);
 }
}



###############################################################################
# Name: edit_window
#
# Purpose: Build the main window.
#
# Role: GUI component
#
{ package edit_window;
  sub new{
        # Blessings & Parameters
        my $self = {}; # Self
        bless($self);
        my ($configuration, $parentwindow, $modifyexistingentry, $ou, %fields) = @_; #
Parameters
        bless($configuration);
```

```
# Variables
my $ldapsearch = $configuration->{LDAPSEARCH};
my $binddn = $configuration->{BINDDN};
my $bindpw = $configuration->{BINDPW};
my $searchdn = $configuration->{SEARCHDN};


# Build dialog box
my $dialog = new Gtk::Dialog();
$dialog->set_modal($true);
$dialog->set_title("Edit");
$dialog->set_transient_for($parentwindow);
$dialog->signal_connect( "delete_event", \&cancelbutton_clicked, $dialog );

# Create a table box
#my $table = new Gtk::Table($#fields, 2, $false);
my $table = new Gtk::Table(23, 2, $false);
$dialog->vbox()->pack_start($table, $true, $true, 4);
$table->show();

# Create rows in editor
my $i = 0;
foreach my $rowname (keys %fields)
{
        # Create label
        my $label = new Gtk::Label( $rowname );
        $table->attach_defaults($label, 0+0, 0+1, $i, $i+1);
        $label->show();

        # Create the line for the fields
        if ($rowname eq $configuration->{PASSWORD_FIELD} ){
                # Is it a password field?  This is special.
                my $hbox = new Gtk::HBox();
                $table->attach_defaults($hbox, 1+0, 1+1, $i, $i+1);
                $hbox->show();

                my $label = new Gtk::Entry();
                if (($fields{$rowname} eq $rowname)
                        or (not $fields{$rowname})){
```

```
                $label->set_text("");
            } else {
                $label->set_text($fields{$rowname});
            }
            $label->set_editable($false);
            $label->can_focus($false);
            $hbox->pack_start($label, $true, $true, 0);
            $label->show();

            my $button = new Gtk::Button("Set password");
            $hbox->pack_start($button, $false, $false, 0);
            $button->signal_connect('clicked',
                \&passwordbutton_clicked,
                $dialog, $label, $rowname, \%fields);
            $button->show();
        }
        else {
            # Is it a normal field?
            my $entry = new Gtk::Entry();
            $table->attach_defaults($entry, 1+0, 1+1, $i, $i+1);
            if($fields{$rowname}){
                $entry->set_text($fields{$rowname});
            }
            $entry->signal_connect('changed', \&entry_changed,
                $rowname, \%fields);
            $entry->show();
        }

        # Bookkeeping
        $i++;
}

# Create "ok" button
my $okbutton = new Gtk::Button("ok");
$dialog->action_area()->add($okbutton);
$okbutton->show();
$okbutton->signal_connect( 'clicked', \&okbutton_clicked,
        $dialog, $modifyexistingentry, $ou, \%fields);


# Create "cancel" button
```

```
	my $cancelbutton = new Gtk::Button("Cancel");
	$dialog->action_area()->add($cancelbutton);
	$cancelbutton->show();
	$cancelbutton->signal_connect( 'clicked', \&cancelbutton_clicked, $dialog);


	# Finish
	$dialog->show();

	# Cleanup
	return $self;
}

###############################################################################
sub entry_changed{
	my ($widget, $tag, $hash) = @_;
	$$hash{$tag} = $widget->get_text();
}

###############################################################################
sub passwordbutton_clicked{
	# Parameters
	my ($widget, $parentwindow, $label, $tag, $hash) = @_;

	# Variables
	my $password;
	my $passwordcheck;

	# Prompt for password
	if (not main::input_box("Enter a password.", \$password,
			$false, $parentwindow)) {
		return;
	}

	# Check password
	if (not main::input_box("Confirm password.", \$passwordcheck,
			$false, $parentwindow)) {
		return;
	}

	# Do the passwords match?
```

```
	if (not ($password eq $passwordcheck)){
		main::error_box("Passwords do not match!", $parentwindow);
		return;
	}

	# Encrypt and save the password
    my $cryptpassword = crypt($password, "b4");
	$cryptpassword = "{CRYPT}$cryptpassword";
	$$hash{'userPassword'} = $cryptpassword;

	# Fancy GUIfication
	$label->set_text($cryptpassword);


}

################################################################################
sub okbutton_clicked{
	# Parameters
	my ($widget, $dialog, $modifyexistingentry, $ou, $hash) = @_;

	# Variables
	my $manager_dn = $configuration->{MANAGER_DN};
	my $base_dn = $configuration->{SEARCHDN};
	my $manager_pw;

	# Find the Manager password?
	if (not main::input_box("Password for $manager_dn.",
		\$manager_pw, $false, $dialog))
	{ return; }


	# Are the field values sane?
	if (not sanitycheck($dialog, $hash)){
		error_box("Invalid field values!", $dialog);
	}

	# Compose ldapadd session
	my $executable;
	if ($modifyexistingentry){
		$executable = $configuration->{LDAPMODIFY};
```

```
        }
        else {
                $executable = $configuration->{LDAPADD};
        }
        my $command = "$executable -v -x -D \'$manager_dn\' -w \'$manager_pw\'";
        #print $command;

        # Start a conversation with ldapadd
   open( LDAPADD, "| $command") or die "could not execute $command\n";

        # Compose ldif-formatted information
        print LDAPADD "dn: $$hash{'dn'}\n";
        # Is it a person?
        if ($ou eq $configuration->{OU_PEOPLE} )
        {
                # Create person object
      print LDAPADD "objectClass: top\n";
      print LDAPADD "objectClass: inetOrgPerson\n";
      print LDAPADD "objectClass: account\n";
      print LDAPADD "objectClass: posixAccount\n";
                foreach my $key (keys %$hash){
                        if($key eq "dn") {
                        }
                        elsif ($key eq "automountInformation" ){
                        }
                        elsif (($key eq "userPassword") and not ($$hash{$key} =~ /CRYPT/)){
                        }
                        elsif ($$hash{$key}) {
                                print LDAPADD "$key: $$hash{$key}\n";
                        }
                }
                print LDAPADD "\n";

                # Create automount object
                if ($$hash{'automountInformation'}){
                        print LDAPADD "dn: cn=$$hash{'uid'},ou=automount,$base_dn\n";
                        print LDAPADD "objectClass: automount\n";
                        print LDAPADD "cn: $$hash{'uid'}\n";
                        print LDAPADD "automountInformation:
$$hash{'automountInformation'}\n";
                        print LDAPADD "\n";
```

```
            }
    }
    elsif ($ou eq $configuration->{OU_GROUP} )
    {
            # Create group object
print LDAPADD "objectClass: top\n";
print LDAPADD "objectClass: posixGroup\n";
            foreach my $key (keys %$hash){
                    if((not $key eq "dn") and ($$hash{$key})  ){
                            print LDAPADD "$key: $$hash{$key}\n";
                    }
            }
            print LDAPADD "\n";
    }
    elsif ($ou eq $configuration->{OU_HOSTS} )
    {
            # Create host object
print LDAPADD "objectClass: top\n";
print LDAPADD "objectClass: ipHost\n";
print LDAPADD "objectClass: device\n";
            foreach my $key (keys %$hash){
                    if((not $key eq "dn") and ($$hash{$key})  ){
                            print LDAPADD "$key: $$hash{$key}\n";
                    }
            }
            print LDAPADD "\n";
    }
    # Is it another kind of object?
    else
    {
            foreach my $key (keys %$hash){
                    if((not $key eq "dn") and ($$hash{$key})  ){
                            print LDAPADD "$key: $$hash{$key}\n";
                    }
            }
            print "\n";
    }

    # End conversation with ldapadd
    close( LDAPADD );
```

```
        # Close this dialog box
        $dialog->destroy();
 }
 sub sanitycheck{
        my ($dialog, $hash) = @_;
        print "FIXME: Sanity Check hard-wired pass\n";
        return $true;
 }

 ###########################################################################
 sub cancelbutton_clicked{
        # Parameters
        my ($widget, $dialog) = @_;

        # Debug
        print "edit_window::cancelbutton_clicked()\n";

        # Do something
        $dialog->destroy();
 }
}



############## Functions ##############

###########################################################################
sub error_box {
        # Variables
        my $okdone = $false;

        # Parameters
        my ($errormessage, $parentwindow) = @_;

        # Build dialog box
        my $dialog = new Gtk::Dialog();
        $dialog->set_modal($true);
        $dialog->set_title("Error");
        $dialog->set_transient_for($parentwindow);

        # Error message text
```

```
	my $label = new Gtk::Label($errormessage);
	$dialog->vbox()->pack_start($label, $false, $false, 10);
	$label->show();

	# OK Button
	my $okbutton = new Gtk::Button("Ok");
	$dialog->action_area()->add($okbutton);
	$okbutton->show();
	$okbutton->signal_connect( 'clicked',
		sub {
			my ($widget, $dialog, $done) = @_;
			$$done = $true;
			$dialog->destroy();
		},
		$dialog, \$okdone);

	# Show dialog box
	$dialog->show();


	# Wait for dialog box to be done
	while (not $okdone){
		Gtk->main_iteration_do($false);
	}
}


################################################################################
sub input_box {
	# Variables
	my $okdone = $false;
	my $canceldone = $false;

	# Parameters
	my ($question, $returnvalue, $visibility, $parentwindow) = @_;

	# Build dialog box
	my $dialog = new Gtk::Dialog();
	$dialog->set_modal($true);
	$dialog->set_title("Question");
	$dialog->set_transient_for($parentwindow);
```

```
# Error message text
my $label = new Gtk::Label($question);
$dialog->vbox()->pack_start($label, $false, $false, 4);
$label->show();

# Input Box text
my $entry = new Gtk::Entry();
$dialog->vbox()->pack_start($entry , $false, $false, 4);
$entry->signal_connect( 'changed',
        sub{
                my ($widget, $returnvalue) = @_;
                $$returnvalue = $widget->get_text();
        },
        $returnvalue);
$entry->set_visibility($visibility);
$entry->grab_focus();
$entry->show();

# OK Button
my $okbutton = new Gtk::Button("Ok");
$dialog->action_area()->add($okbutton);
$okbutton->show();
$okbutton->signal_connect( 'clicked',
        sub {
                my ($widget, $dialog, $flag) = @_;
                $$flag = $true;
                $dialog->destroy();
        },
        $dialog, \$okdone);
$okbutton->grab_focus();

# cancel button
my $cancelbutton = new Gtk::Button("Cancel");
$dialog->action_area()->add($cancelbutton);
$cancelbutton->show();
$cancelbutton->signal_connect( 'clicked',
        sub {
                my ($widget, $dialog, $flag) = @_;
                $$flag = $true;
                $dialog->destroy();
```

```
                },
                $dialog, \$canceldone);

        # Show dialog box
        $dialog->show();

        # Wait for dialog box to be done
        while ((not $okdone) and (not $canceldone)){
                Gtk->main_iteration_do($false);
        }

        # Return something intelligent
        if ($okdone){
                return $true;
        }
        else {
                $$returnvalue = "";
                return $false;
        }
}


################################################################################
# FUNCTION: finger
#
# PURPOSE: Ask the finger-server about a particular user
#
# PARAMETERS:
#       $finger_server - the name of the finger server
#       $key - the username that we're looking for.
#
# RETURNS: A hash with the fields from the finger server
#
sub finger
{
      # Parameters
      my ($finger_server, $key) = @_;
         my %ret;

         print "finger($finger_server, $key)\n";
```

```
# Protections
if (not $finger_server){ return };

# Finger
print "---------- Open \<FINGER\> ----------\n";
open(FINGER, "finger -smp $key\@$finger_server |");
while( <FINGER> ) {
	# Parse input
	if( $_ =~ /exact match found/) {
			# Did we find a record?
			# Grab the name from the header
			my $val = <FINGER>;
		$val =~ s|^ *\"||g; #"# Chop off leading garbage
		$val =~ s| *\(.*$||g; # chop off trailing garbage
		chomp($val); # chop off more trailing garbage"

			# store into pre-known slot
			$ret{'FullName'} = $val;

			# debug
			print "\$ret{FullName} = $ret{'FullName'}\n";

	}
	elsif( $_ =~ /\:/ ) {
			# Get field name
			my $fieldname = $_;
		chomp($fieldname);
			$fieldname =~ s|[\:\-\ ]||g;

			# Get the first value
			my $val = <FINGER>;
			chomp $val;
			$val =~ s|^ *||g;
			$ret{$fieldname} = $val;

			# Debug
			print "\$ret{$fieldname} = $ret{$fieldname}\n";
	}
	else{
		# Print anything we didn't understand
		#print "x: $_";
```

```
            }
        }
        close( FINGER );
        print "---------- Close \<FINGER\> ----------\n";

            # Finish up
            return %ret;
}
```