Creating a Shared Address Book with LDAP

Often it's useful to create a shared directory, rather than relying on individual address books to address mail. The best way to do this these days is with LDAP, the Lightweight Directory Access Protocol.

A directory is a database that is mostly used for lookups, with only occasional updates. It's accordingly optimized for read access, rather than write.

In this article, we'll implement a basic LDAP server, used only for email address lookups. LDAP can also be used to implement a complete authentication and authorization system - essentially a single-sign-on system which simplifies user administration across multiple machines, but the setup for that is rather more complex, and for those who are interested, the address book implementation is a good way to learn the basics of LDAP.

For this exercise, I used Fedora Core 4, but the same techniques should work on any Linux distribution, perhaps with some changes to the paths of various files. In particular, we're using the OpenLDAP server, which is a core component of many distributions.

Setting Up the LDAP Server

If your system does not already have the `slapd`
server installed, you will have to install it. The specifics here vary between distributions - on Fedora Core, it is simply a matter of giving the command "`yum install ldap-servers`".

The next step is to configure the LDAP server by editing the file `/etc/openldap/slapd.conf`. We only need to change a couple of things in here. Nominally, the address book serves a domain, and we need to set the suffix entry for that; and we also need to set the DN (Distinguished Name) and password hash for the administrator or manager account. I chose to use the `lesbell.org`
domain, as I own that one, but you can use almost anything you want - you can make up a domain name like "`ourfamily.pvt`", if you want. Here's the relevant section from the `slapd.conf` file, with the original entries commented out but left as examples:

```
#######################################################################
# ldbm and/or bdb database definitions
#######################################################################

database        bdb
#suffix         "dc=my-domain,dc=com"
suffix          "dc=lesbell,dc=org"
#rootdn         "cn=Manager,dc=my-domain,dc=com"
rootdn          "cn=root,dc=lesbell,dc=org"
# Cleartext passwords, especially for the rootdn, should
# be avoided.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw                  secret
# rootpw                {crypt}ijFYNcSNctBYg
```

In this example, we are using *simple authentication* - this means sending passwords as plain text over the network. This is OK for a small office or family setup, but in less trusted environments - and particularly when using LDAP for authentication itself - I would configure LDAP to use SSL/TLS encryption or Cyrus SASL (Simple Authentication and Security Layer). However, that topic is an article - or more - in its own right.

Once you have saved this file, you can start the LDAP server (`slapd`):

```
[root@loki ~]# service ldap start
Checking configuration files for slapd:  config file testing succeeded
                                                           [  OK  ]
Starting slapd:                                             [  OK  ]
[root@loki ~]#
```

It's a good idea to test that the server is running correctly with a command-line request like this one:

```
[root@loki ~]# ldapsearch -x -b '' -s base '(objectclass=*)' namingContexts
# extended LDIF
```

```
#
# LDAPv3
# base <> with scope base
# filter: (objectclass=*)
# requesting: namingContexts
#

#
dn:
namingContexts: dc=lesbell,dc=org

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
[root@loki ~]#
```

So far, so good. To make sure the server will automatically restart upon reboot, use the command:

```
[root@loki ~]# chkconfig ldap on
[root@loki ~]#
```

Creating the Address Book

The next step is to create an LDAP entity for the Base Distinguished Name (DN) of the address book. I'm going to continue with the lesbell.org domain, but you should substitute your own wherever required. To create the base DN entry in the LDAP directory, we'll create an LDIF file and import it with the `ldapadd` command (don't worry - adding entries is much easier later). LDIF is the LDAP Interchange Format, a text file format which all LDAP servers can export and import. It's a little verbose, but by using it we'll get to see a little of the logic of how the LDAP server works, which is useful.

An LDAP directory contains a *Directory Information Tree*, and since yours is currently empty, we'll need to create a couple of branches. At the top, you'll need to create an entry for your organisation (in my example, lesbell.org, but use your own, obviously). Then, underneath that, you'll need to create an entry for the address book itself, which is - by convention - called "People". Each entry in the LDIF file consists of multiple lines, with blank lines between entries. Each entry starts with a Distinguished Name (dn), which uniquely identifies the entry. If you have experience with relational databases, think of the dn as being like a primary key. The DN typically is made of several Domain Components (dc), which are often - but not always - the parts of the organisation's Internet domain name (for example, a DN could be something like o=ACP,l=Sydney,c=AU).

An LDAP directory entry must belong to at least one *objectClass*, and it's quite common for entries to belong to multiple ObjectClasses, as you'll see. Each objectClass defines a set of attributes that are either mandatory or optional for that entry. For example, our first entry will be an `organization`, and a necessary attribute for an `organization` is an `organizationName` (mercifully abbreviated to o).

First, use your favourite editor to create a file, which we'll call `basedn.ldif`:

Listing 1 - basedn.ldif

```
dn:     dc=lesbell,dc=org
objectClass:    top
objectClass:    dcObject
objectClass:    organization
dc:     lesbell
o:      Les Bell and Associates Pty Ltd

dn:     ou=People,dc=lesbell,dc=org
objectClass:    top
objectClass:    organizationalUnit
ou:     People
```

The second part of this file creates a "People" Organizational Unit - this is a branch of the LDAP directory tree which

will contain all the addressbook entries (LDAP directories can also contain machines, services and other stuff often associated with DNS). Once you have saved the file, import it into the LDAP database with the `ldapadd` command:

```
[root@loki ldap]# ldapadd -x -D 'cn=root,dc=lesbell,dc=org' -W -f basedn.ldif
Enter LDAP Password:
adding entry "dc=lesbell,dc=org"

adding entry "ou=People,dc=lesbell,dc=org"

[root@loki ldap]#
```

(Here, the LDAP password is the rootpw from the slapd.conf file - in this case, "secret").

Everything seems to have worked, but a quick query should confirm this:

```
[root@loki ldap]# ldapsearch  -x -b 'dc=lesbell,dc=org' '(objectclass=*)'       #
extended LDIF
#
# LDAPv3
# base <dc=lesbell,dc=org> with scope sub
# filter: (objectclass=*)
# requesting: ALL
#

# lesbell.org
dn: dc=lesbell,dc=org
objectClass: top
objectClass: dcObject
objectClass: organization
dc: lesbell
o: Les Bell and Associates Pty Ltd

# People, lesbell.org
dn: ou=People,dc=lesbell,dc=org
objectClass: top
objectClass: organizationalUnit
ou: People

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
[root@loki ldap]#
```

If your ldapadd command doesn't seem to have worked, note any error messages, then go back and edit your LDIF file to fix the problem. Since this command imports two entries, it's possible that it might have accepted the first entry but not the second, in which case, you should delete the first entry from the file before fixing the second entry - otherwise the `ldapadd`
command will complain that it cannot add the first entry since it already exists. If the command successfully imports the entries but you subsequently notice an error, you can edit the file and then use command line recall and editing to change the `ldapadd` command into an `ldapmodify` command which will amend the entry.

Adding a First Entry

We'll continue working at the command line to add a first entry in the address book. Table 1 shows common attributes used in address book-type applications. If you want to use any of thse attributes, you'll need to specify the corresponding objectClass that contains that attribute. You'll notice that a lot of the useful attributes come from the `inetOrgPerson` and `organizationalPerson` objectClasses, so most people entries will use those as well as the basic `person` objectClass. These are all defined in schema files which you'll probably find in `/etc/openldap/schemas`, if you want to go exploring.

Generally, LDAP servers allow anonymous connection to the directory - technically known as *anonymous binding*. However, you might want to allow users to edit their own entries in larger environments, and in that case, the server

needs to know the identity of the user binding to it, and this means setting a password for authentication purposes. This is stored in the LDAP directory itself - anyone who wants to bind to the directory will have to have an entry, and that entry will have to have a `userPassword`
attribute. More conventional directory entries - for contacts, rather than directory users - will not have a userPassword attribute, however.

Other attributes include various forms of address information and phone numbers; many other attributes are supported, and you are encouraged to read the schema files to discover them. By way of example, here is the definition of an organizationalPerson:

```
objectclass ( 2.5.6.7 NAME 'organizationalPerson'
       DESC 'RFC2256: an organizational person'
       SUP person STRUCTURAL
       MAY ( title $ x121Address $ registeredAddress $ destinationIndicator $
preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $
telephoneNumber $ internationaliSDNNumber $ facsimileTelephoneNumber $
street $ postOfficeBox $ postalCode $postalAddress $
physicalDeliveryOfficeName $ ou $ st $ l ) )
```

As you can see, there are many more optional ('MAY') attributes.

Table 1 - LDAP Attributes Used in Address Book Entries

Attribute	ObjectClass	From Schema File	Comment
commonName, cn	person	core.schema	Person's full name
givenName, gn	inetOrgPerson	inetorgperson.schema	First name
surname, sn	person	core.schema	Last or family name
uid	inetOrgPerson	inetorgperson.schema	User ID - used for logon authentication (see notes on phpldapadmin below)
userPassword	person	core.schema	A password is required if the user will want to bind to the LDAP server.
mail	inetOrgPerson	inetorgperson.schema	RFC2822 email address
organizationName, o	inetOrgPerson	inetorgperson.schema	Company or organization name
postOfficeBox	organizationalPerson	core.schema	Post office box
postalAddress	organizationalPerson	core.schema	Street address
localityName,l	organizationalPerson	core.schema	City
st	organizationalPerson	core.schema	State
postalCode	organizationalPerson	core.schema	Postcode
telephoneNumber	inetOrgPerson	inetorgperson.schema	Work phone number
facsimileTelephoneNumber	inetOrgPerson	inetorgperson.schema	Fax number
mobile	inetOrgPerson	inetorgperson.schema	Mobile
homePhone	inetOrgPerson	inetorgperson.schema	Home phone number

Armed with this information, you should now be able to create a file - call it `self.ldif` - containing your own personal information. I've shown my own in Listing 2. If you want to be able to bind to the LDAP server and edit your own entry, remember to add a userPassword.

Listing 2 - LDIF File to add a person

```
dn:     cn=Les Bell, ou=People,dc=lesbell,dc=org
objectClass:    top
objectClass:    person
objectClass:    inetOrgPerson
objectClass:    organizationalPerson
```

```
cn:     Les Bell
gn:     Les
sn:     Bell
userPassword:   secret
mail:   lesbell@lesbell.com.au
o:      Les Bell & Associates Pty Ltd
postofficebox:  PO Box 173
l:      Frenchs Forest
st:     NSW
postalCode:     1640
countryName:    Australia
telephoneNumber:        (02) 9451 1144
facsimileTelephoneNumber:       (02) 9451 1122
mobile: 0408 239 711
```

You should now be able to add this, with a command like the following:

```
[les@loki ldap]$ ldapadd -x -D 'cn=root,dc=lesbell,dc=org' -f self.ldif -W
Enter LDAP Password:
adding new entry "cn=Les Bell, ou=People,dc=lesbell,dc=org"

[les@loki ldap]$
```

The options here are:

- `-x` to use simple authentication, rather than SASL
- `-D 'cn=root,dc=lesbell,dc=org'` to specify the DN we're binding with - this must match the `rootdn` entry in `slapd.conf`
- `-f self.ldif` to specify the file to be imported, and
- `-W` to prompt for the simple authentication password - which should match the `rootpw` entry in `slapd.conf`

To test that the entry was accepted, try a simple search on the givenName of the entry:

```
[les@loki ldap]$ ldapsearch -x -b "dc=lesbell,dc=org" "(gn=Les)"
# extended LDIF
#
# LDAPv3
# base <dc=lesbell,dc=org> with scope sub
# filter: (gn=Les)
# requesting: ALL
#

# Les Bell, People, lesbell.org
dn: cn=Les Bell,ou=People,dc=lesbell,dc=org
physicalDeliveryOfficeName: Les Bell & Associates Pty Ltd
postalAddress: PO Box 173
objectClass: top
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
cn: Les Bell
givenName: Les
sn: Bell
userPassword:: c2VjcmV0
mail: lesbell@lesbell.com.au
o: Les Bell & Associates Pty Ltd
postOfficeBox: PO Box 173
l: Frenchs Forest
st: NSW
postalCode: 1640
telephoneNumber: (02) 9451 1144
facsimileTelephoneNumber: (02) 9451 1122
mobile: 0408 239 711

# search result
```

```
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
[les@loki ldap]$
```

It looks as though this example worked. The next step is to configure an email client and check that we can do LDAP lookups.

Configuring Thunderbird

Thunderbird is extremely easy to configure to use your new LDAP directory. After starting Thunderbird, select Edit -> Preferences from the application menu; this will bring up the Preferences dialog window. Now, click on the "Composition" icon in the list at left, and check "Directory Server", then click on "Edit Directories...". In the LDAP Directory Servers dialog, click on add, and fill in the details of your new server.

Table 2 - The Thunderbird "Directory Server Properties" Dialog

Field	Explanation	Example(s)
Name:	A name that represents your LDAP directory	LDAP
Hostname:	The hostname or IP address of the LDAP server	`locahost, sleipnir, 192.168.0.1`
Base DN:	The Distinguished Name of the part of the LDAP directory information tree that you want queries to search.	`ou=People,dc=lesbell,dc=org`
Port number:	The port number the LDAP server listens on.	`389`
Bind DN:	A distinguished name that you want to use to connect to the LDAP server. The corresponding entity should have a userPassword attribute, as shown in our example	`cn=Les Bell,dc=lesbell,dc=org`

Once you have set this up, the LDAP server should appear in your address book. If you select it and start typing a name into the "Name or Email" field, it should list matching LDAP entries, as shown in Figure 1

.Thunderbird does not currently support editing of LDAP directory entries (see https://bugzilla.mozilla.org/show_bug.cgi?id=86405), so - unless you want to create an LDIF file containing all your contacts information - you will probably want to install a graphical LDAP editor of some kind.

If you already have a large collection of email addresses in a Thunderbird address book, you can export them with the Tools -> Export menu option, which will save them in an LDIF format. However, the resultant LDIF contains some

attributes which are from a non-standard and out-of-date schema, and you will need to process the file before importing it. You will need to fix up the dn: entries, which are based on a "cn=,mail=" format, and replace the "mail=" part of the dn with the correct value for your setup, as well as removing some lines and tidying up some misplaced quote marks.

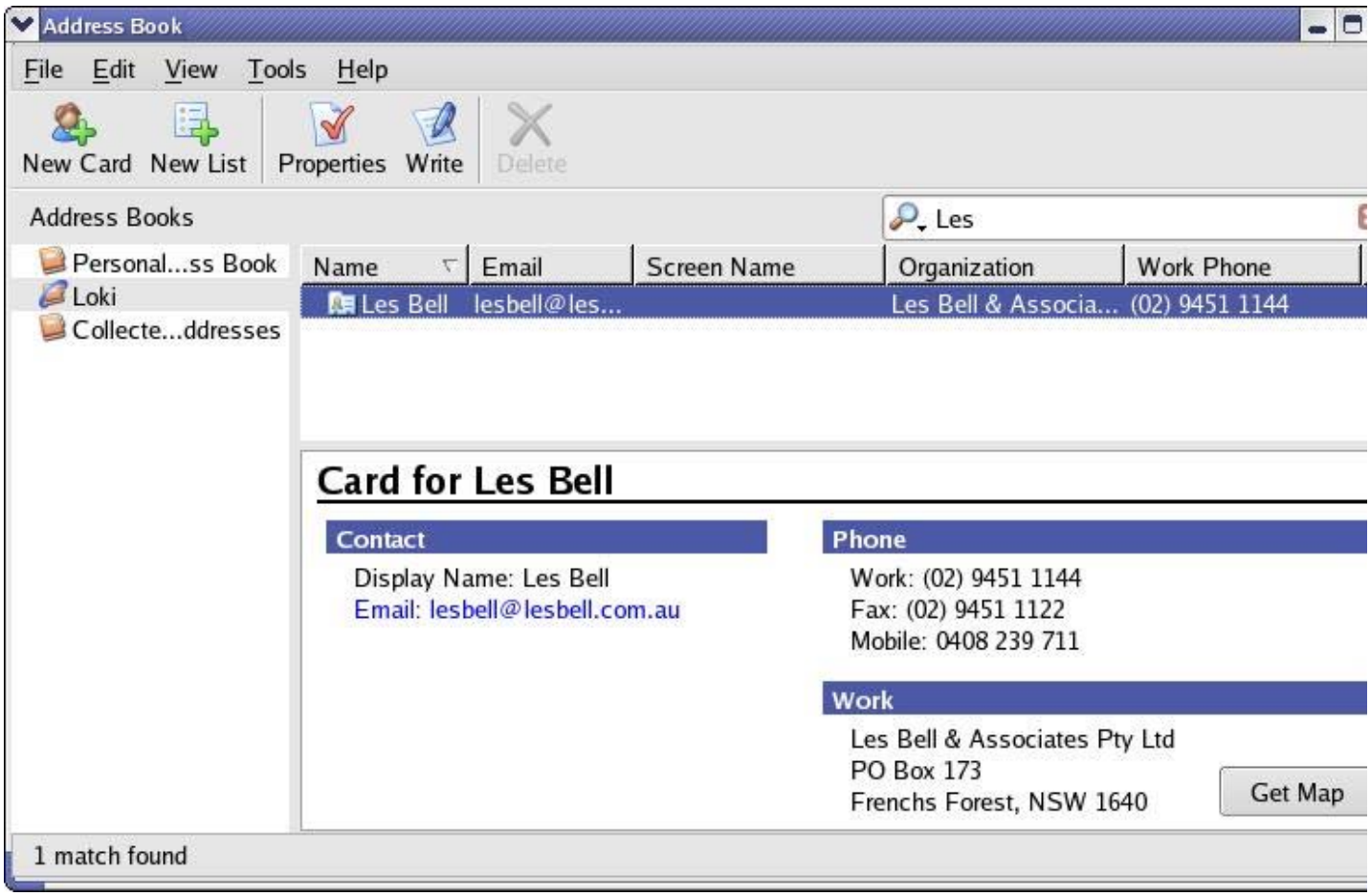
You can do this with a simple Perl script:

```
#!/usr/bin/perl -pi
s/dn: cn=\"(.*)\"/dn: cn=$1/;
s/mail=.*/ou=People,dc=lesbell,dc=org/;
s/givenName: \"/givenName: /;
s/cn: \"(.*)\"\n/cn: $1/;
s/sn: (.*)\"\n/sn: $1\n/;
s/modifytimestamp.*\n//;
s/xmozillanickname.*\n//;
s/objectclass: mozillaAbPersonObsolete/objectclass: mozillaOrgPerson/;
```

Mark this script as executable, then run it with these commands:

```
chmod +x tbirdfix
./tbirdfix addressbook.ldif
```

You will also need to obtain the mozillaOrgPerson schema from http://www.linux.com/blob.pl?id=02a29e9f755b27c40c3ae45acb69c5ee and save it in your `/etc/openldap/schema` directory, then add it into the list of schemas at the top of `/etc/openldap/slapd.conf`. After restarting slapd, you will be able to import your exported address book LDIF file.



Other Email LDAP Clients

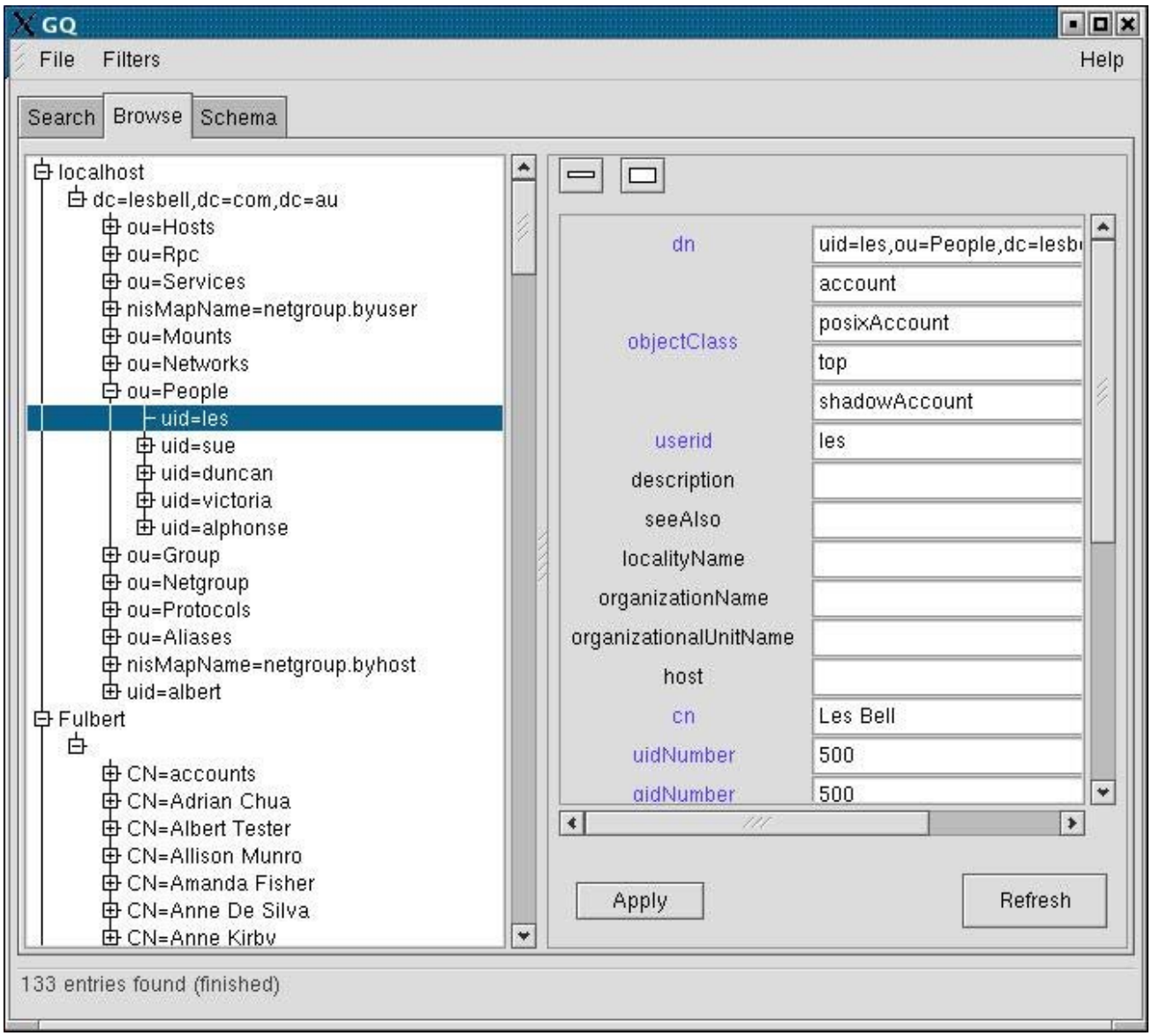
Other email programs can similarly be configured as LDAP clients, including Microsoft Outlook, Eudora and many others. In all cases, configuration is very similar to that for Thunderbird. Note that a Bind DN is usually not required, unless you intend to allow people to edit entries.

LDAP Editors

The good news is that there are lots of good LDAP editors and browsers available, and these are particularly useful tools if you want to learn more about LDAP, as well as making it easy to maintain your LDAP address book.

GQ

The most commonly-encounted LDAP client on Linux is called gq. This has a simple interface, and is suitable for basic testing and exploration of LDAP concepts. Its Browse tab is particularly powerful, but gq is not being actively maintained at the moment.



Directory Administrator

For administering LDAP servers used for authentication, I prefer to use Directory Administrator. This is a user-friendly application that has wizards to step the user through creation of users and groups, and can automatically set up support for logon from Windows clients via a Samba server. While it can be used for maintenance of an address book, it tends to demand setting of attributes that are more appropriate to authentication applications, such as userPassword. Installation on Fedora Core is a snap:

```
yum install diradmin
```

phpldapadmin

This is a browser-based utility for managing LDAP directories. Like Directory Administrator, it has much more sophisticated features than are required here, but it seems to be more versatile and easier to use for the simple address book application. It also has the advantage that, being browser-based, any user can access it across the network, and it is platform-independent. Installation on Fedora Core is simple, with the command:

```
yum install phpldapadmin
```

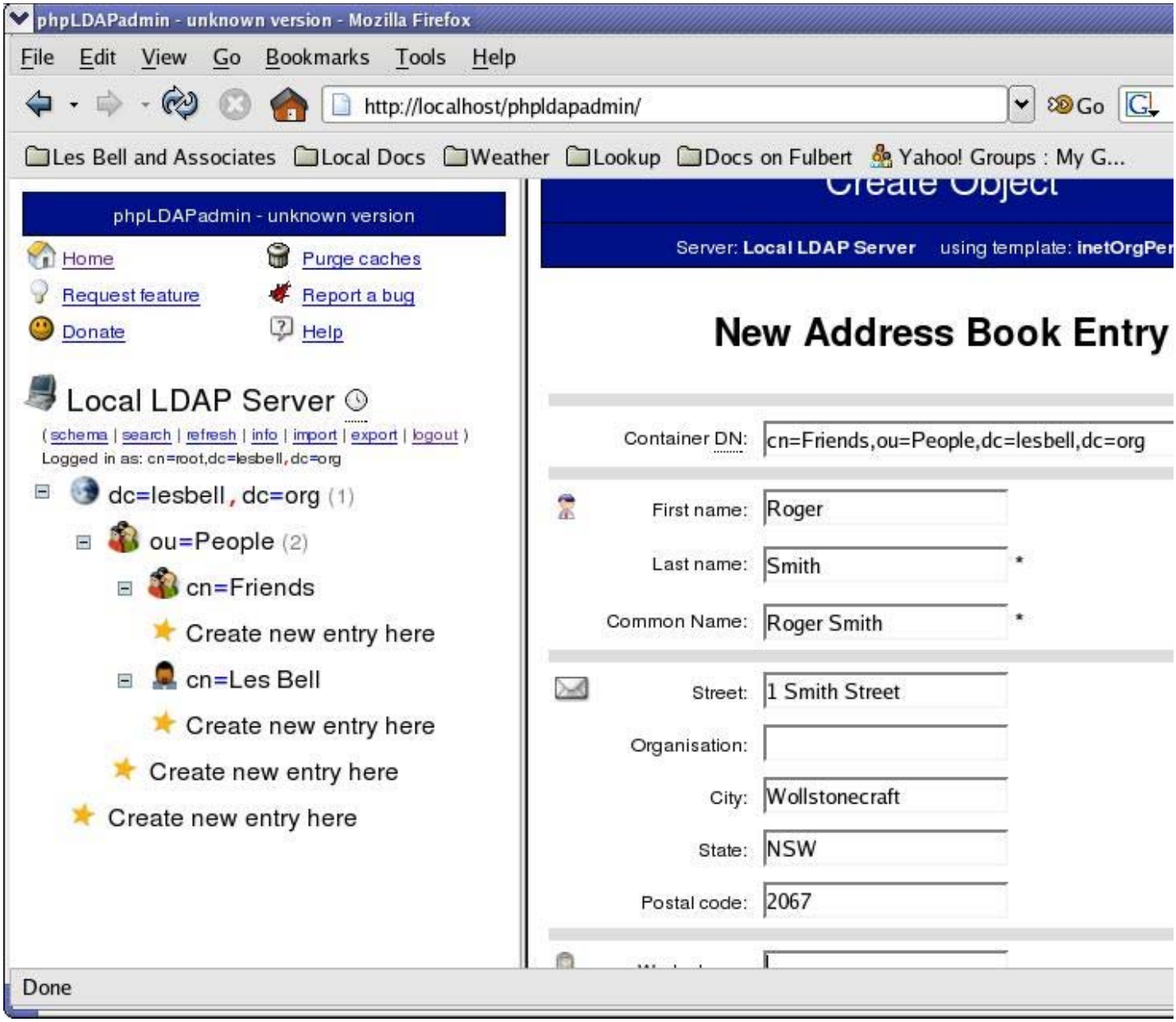
By default, phpldapadmin wants you to log in using a uid and userPassword, so if you want users to be able to use it, you should give them a uid field. If you have not already set this up, you can edit /etc/phpldapadmin/config.php and find the line that reads:

```
$ldapservers->SetValue($i,'login','attr','uid');
```

then modify it to read

```
$ldapservers->SetValue($i,'login','attr','dn');
```

This will alllow you to log in with a DN like `'cn=root,dc=lesbell,dc=org'` and the corresponding password.



Other LDAP administration/browsing tools include GOsa and JXplorer - the latter is a Java application, and can be run on Windows as well as on Linux. Download and documentation links are in the References below.

Where Next?

If you've got all the steps in this introduction under your belt, you might like to tackle some more advanced functionality. For example, Thunderbird supports a more sophisticated schema, which you can download from the links in the references below. Just rename the file to mozillaAbPerson.schema and place it in the same directory as the other schemas (e.g. /etc/openldap/schemas). Then edit slapd.conf and add a line to include the new schema

```
include mozillaAbPerson.schema
```

This will make all the new fields accessible, and you'll find that PhpLdapAdmin already understands this schema.

From here, the next upgrade is to enable SSL support, so that all data is encrypted. This is a slightly involved procedure, which requires you to generate an SSL certificate using OpenSSL and then install it, but there is docimentation on the OpenLDAP web site.

In summary, setting up an LDAP server provides useful functionality in offices where people need access to shared address books, and is an interesting exercise in learning the basics of LDAP.

Tech Terms

LDAP - The Lightweight Directory Access Protocol is used to connect clients, such as email programs, to directory servers, in order to perform lookups of data such as email addresses

DN: Distinguished Name - a unique string which identifies an object in the LDAP directory information tree. It is composed of a comma-separated sequence of Relative Distinguished Names (RDN's).

SSL/TLS: Secure Sockets Layer / Transport Layer Security. A technique for encrypting traffic at the transport layer to provide protection against sniffing. Often used between web servers and browsers (when the little padlock turns gold) but also used to encrypt LDAP sessions as well as email protocols such as IMAP.

References and Further Reading

The OpenLDAP web site: http://www.openldap.org

Carter, Gerald, "LDAP System Administration", O'Reilly, 2003. ISBN 1-56592-491-6

"LDAP Support in Mozilla Thunderbird", available online at http://www.mozilla.org/projects/thunderbird/specs/ldap.html

A schema for mozillaAbPerson suitable for use with Thunderbird 1.0 can be found at http://www.netpress.com/mozilla/ab2ldap_1/mozilla_op20.schema. For Thunderbird 1.5RC1 and later, use "LDAP Address Book Schema - Alpha Version", available online at http://wiki.mozilla.org/MailNews:LDAP_Address_Books

Directory Administrator: http://diradmin.open-it.org/index.php

GQ: http://biot.com/gq/

PhpLDAPadmin: http://wiki.pldapadmin.com

GOsa: https://gosa.gonicus.de/

JXplorer: http://jxplorer.org/

Page last updated: 04/Jul/2006 **Back to Home**

.....