LDAP Filters and Searching LDAP Directories — Solutions

1 Background:

A grammar is a notation that defines the syntax of a computer language. You have already seen an example of a grammar on slides 22 and 23 of Lecture8 (about Interaction Diagrams) in your OOT notes, defining the rules for the message label syntax of collaboration diagrams.

Today we look at the syntax of LDAP filters, a simple standard language used in defining searches of an LDAP directory. It corresponds loosely to the SQL SELECT statement, except that it is very standardised. We start by looking at a grammar that defines the syntax of LDAP filters.

1.1 A Grammar that Defines the Syntax of LDAP Filters

The rules for the grammar are given in RFC 822. You can find this RFC (and the others) at [http://www.faqs.org/rfcs/rfc822.html](http://www.faqs.org/rfcs/rfc822.html). Let me summarise the description of the grammar from the RFC for you.

- **a rule** is given by a name such as filter, filtercomp,... Eventually a rule is further defined until it consists of literal values, and values such as AttributeDescription or AttributeValue, both defined in RFC 2251.

- **a literal value** is put in double quotes, such as "(" , ")" , "=" , ">=" , ">*"... below. A literal value is simply typed into the filter just the way it is written; from this, we see that every filter is always enclosed in parentheses (see the examples in section 1.2 on page 3).

- **alternatives** are separated by a slash "/". For example, the rule:

```
item     = simple / present / substring / extensible
```

means that an item is either defined by the rule for simple, or for present, or for substring, or for the rule defining extensible.

- **An optional** item is in square brackets, such as in the rule

```
substring = attr "=" [initial] any [final]
```

where initial and final are optional.

*Please stop telling me you have never seen a grammar before!
LDAP Filters and Searching LDAP Directories
Systems and Network Management

- **grouping** is defined by enclosing the elements in parentheses, so that they are treated as a single element. So “elem (foo / bar) elem)” can match “elem foo elem” and “elem bar elem.” See the definition of the rule any below for another example.

- **repetition** is defined with a “*” appearing before a rule. For example,

  
  
  *filter

  
  means “zero or more repetitions of a filter.” A number in front of the star is a minimum number of repetitions, so

  
  1*filter

  
  means “one or more repetitions of filter.”

  
  Let us look at the definition of any:

  
  any   = "*" *(value "*")

  
  This means that to type in part of an LDAP filter that is defined by any, we type:

  
  - a literal star, i.e., * (in other words, we type \(\text{Shift-8}\))
  
  - Next we type zero or more repetitions of:
    
    - a legal attribute value, followed immediately by
    
    - a star * (in other words, we type \(\text{Shift-8}\))

  The following lines contain text that matches this definition of any:

  
  *
  
  *1*
  
  *this*
  
  *this*that*
  
  *1*2*3*4*5*6*7*8*9*10*11*12*13*14*

  
  Note that an AttributeValue cannot be empty, or contain an unquoted star *.

  An important thing to understand is that a grammar only defines the syntax, not the meaning of a computer language.

  Now let us look at the complete grammar itself.

  From /usr/share/doc/openldap-2.0.27/rfc/rfc2254.txt;

  
  filter   = "(" filtercomp ")"
  filtercomp = and / or / not / item
  and      = "+" filterlist
  or       = "|" filterlist
  not      = "!” filter
  filterlist = 1*filter
  item      = simple / present / substring / extensible
  simple    = attr filtertype value
  filtertype = equal / approx / greater / less
  equal     = "="
  approx    = "~~="
1.2 Examples of LDAP Filters

Here are some examples of LDAP filters from rfc2254.txt:

(cn=Babs Jensen)
(!((cn=Tim Howes))
(&(!objectClass=Person)((sn=Jensen)(cn=Babs J*))
(o=univ*of*mich*)

1.3 Using ldapsearch

Here is an example of using ldapsearch to search our LDAP server, shown in the lecture:

ldapsearch -x '(&(acType=STF)(!(year=3)(course=41300)(classcode=W)))' cn

Here are two examples of searching the VTC LDAP server, from the lecture:

ldapsearch -x -h ldap.vtc.edu.hk -b "ou=ICT,ou=TY,o=ftstudent,dc=vtc.edu.hk" \ '(&(acType=STF)(!(year=3)(course=41300)(classcode=W)))' uid

and

ldapsearch -x -h ldap.vtc.edu.hk -b "ou=ICT,ou=TY,o=staff,dc=vtc.edu.hk" \ '(&(acType=STF)(!(year=3)(course=41300)(classcode=W)))' cn

Some points about ldapsearch:

1. You need to use simple authentication with our server. It will not work unless you use the option “-x”.

2. The default host and base for the LDAP server are set in the file /etc/openldap/ldap.conf. They will be set to

   BASE dc=tyict,dc=vtc,dc=edu,dc=hk
   HOST ldap.tyict.vtc.edu.hk

   if you configured your machine correctly with authconfig when you installed Linux.

3. To choose another LDAP server, such as the VTC LDAP server, use the “-h hostname” option, where hostname is the hostname of the LDAP server.
4. To choose a different base DN, use the “-b "base-distinguished-name"” option. Quote the DN, otherwise the shell will interpret the “=” signs.

5. You can authenticate to the LDAP server with a bind operation. To authenticate as yourself using ldapsearch, use the options -D 'uid=your-user-id,ou=People,dc=tyict,dc=vtc,dc=edu,dc=hk' -W

The -W option causes you to be prompted for your password. You need to bind and search at the same time!

1.4 LDAP URLs

The grammar for an LDAP URL is defined by RFC 2255. An LDAP URL has the form:

ldap://host[:port]/base?attr?scope?filter

Here is a (partial) grammar:

ldapurl = ldap://" [hostport] ["/
[[dn ["?" [attributes] ["?" [scope]
["?" [filter] ["?" extensions]]]]

Examples:

ldap://ictlab/ou=People,dc=tyict,dc=vtc,dc=edu,dc=hk?uid?one?(uid=nicku)

1.5 About LDIF

LDIF is the format of text data used to read and write an LDAP server. LDIF has a simple format that you can see as the result of any LDAP search.

The name of each attribute starts a line (no spaces before the attribute name), and is followed by a colon ‘:’ and a single space, then its value. If the value is too long to conveniently fit on one line, it can be “folded” onto more than one line. After a line break, the folded line continues with exactly one space before the continued line.

If the entry contains leading spaces, or any other special characters (for example if it is a photo), then it can be encoded with BASE64 encoding. On a Linux system, you can decode such text by running the program mimencode -u, and then you can copy and paste the encoded text as input to mimencode -u.

2 Procedure

1. Which of the following text from parts of LDAP filters match the definition of any from the grammar for LDAP filters? .................................

(c) and (f) only, since the grammar requires a match to any to begin and end with an asterisk.

If you put the string “o=” in front of each, which ones match the definition of substring? .................................

All of them do, except for (d), since two asterisks together do not match the definition of substring in the grammar.

(a) *John
LDAP Filters and Searching LDAP Directories
Systems and Network Management

(b) John*
(c) *John*
(d) this*is**John*
(e) *is*this*John*
(f) *How*about*this*one*here*

2. For each of the following filters, if you remove the external parentheses, indicate what term in the grammar the result matches, for example, simple, present, substring,...

(a) !(cn=nicku)) ......................... This matches not
(b) (cn=Nick*) ......................... This matches substring
(c) (cn=*) ......................... This matches present

3. (a) Write a filter that shows the LDAP entry for your account on ldap.tyict.vtc.edu.hk. Test it using ldapsearch.

(b) Now repeat the search, but bind to your account using the options -D ⟨yourDN⟩ -W. Use diff to compare the results. Is there any difference in the attributes returned? .................................

Yes. I obtained the output in each case like this:
$ ldapsearch -x ' (uid=nicku) ' > ~/noauth.ldif
$ ldapsearch -x -D 'uid=nicku,ou=People,dc=tyict,dc=vtc,dc=edu, dc=hk' -W ' (uid=nicku) ' > ~/auth.ldif
$ diff -u ~/noauth.ldif ~/auth.ldif
The difference was that my MD5-hashed password was visible when I did the search while binding to the server.

4. Write a filter to select all students in your class.

Test it using ldapsearch; display only the names of the students.

5. Write a filter to select all students in year 2 of the ICT Department.
If we assume that only ICT students are in the directory, then this would do:
(year=2)

However, this is not a safe assumption, so it is better to use the department attribute:
(&department=ICT)(year=2))

Count the number of students that is returned. ........................................

$ ldapsearch -x '(&(department=ICT)(year=2))' dn |
grep '^dn:' | wc -l
371

6. Determine the number of entries at the one level immediately below the base level of the LDAP server ldap.tyict.vtc.edu.hk. In other words, all entries immediately below the DN dc=tyict, dc=vtc, dc=edu, dc=hk.

As Albert taught you so well last year, there are three scopes for searching a directory:
base where you search only the base entry for attributes that match your filter,
one level where you search all entries immediately below the base entry, but not the base entry itself,
sub: a subtree scope results in searching the base entry and all entries below it in the directory tree.
These are selected in the ldapsearch command with the scope option -s:

$ ldapsearch -x -s one dn | grep '^dn:' | wc -l
17

Determine the DN of your entry in both servers, in our server ldap.tyict.vtc.edu.hk:

$ ldapsearch -x -LLL '(uid=nicku)' dn
dn: uid=nicku, ou=People, dc=tyict, dc=vtc, dc=edu, dc=hk

Here is a search for a student:

$ ldapsearch -x -LLL -b 'dc=vtc.edu.hk' -h ldap.vtc.edu.hk
  '(uid=nicku)' dn
dn: uid=nicku, ou=ICT, ou=TY, o=staff, dc=vtc.edu.hk

In both cases, the entry is at at two levels below the base: we have base, then ou=People, then the entry for the person.

and in ldap.vtc.edu.hk:

$ ldapsearch -x -LLL -b 'dc=vtc.edu.hk' -h ldap.vtc.edu.hk
  '(uid=nicku)' dn
dn: uid=nicku, ou=ICT, ou=TY, o=staff, dc=vtc.edu.hk

And here is a search for the same student as above:

$ ldapsearch -x -LLL -b 'dc=vtc.edu.hk' -h ldap.vtc.edu.hk
  '(uid=10954118)' dn
dn: uid=10954118, ou=People, dc=tyict, dc=vtc, dc=edu, dc=hk

Note that in this case, my entry is four levels below the base: base → o=staff → ou=TY → ou=ICT → entry for nicku.
The entry for the student is also four levels below the base:
base → ou=ftstudent → ou=TY → ou=ICT → entry for student.

Is the structure of the directory hierarchical or flat? ..............................
Neither directory tree is totally flat, but the fact that all entries in our directory are two levels below the base, while the level of a user in the VTC directory is four levels down indicates that the VTC directory has been designed with a more hierarchical structure.

Although it seems that this design is more beautiful and it seems as if will be easier to manage, any reorganisation of the VTC will require major changes to the directory. Since a directory is mission critical, it is better to structure the directory so that changes in the organisation will require less radical re-organisation of the directory structure.

With the design of the ICT directory, we can reorganise the directory simply by changing attributes within the affected entries, rather than changing their distinguished names, and changing their relative distinguished names. If we want to introduce a new category that is neither staff nor student (such as alumni for graduates of ICT), then we can simply introduce a new value for an attribute.

The authorities who designed LDAP, created the IETF standards, and wrote the code, said many times that a flatter directory structure is better within the constraints of replicating the directory, of organising referrals from one directory to another, and administrative concerns. I trust them, and perhaps you should too.

Compare this with the VTC LDAP server, ldap.vtc.edu.hk, looking under the base dc=vtc.edu.hk. Is the VTC LDAP server hierarchical or flat in structure?