09. 기술문서 : LDAP의모든것2

This page last changed on 2007년 05월 23 by joon.

LDAP에 대한 모든 것 Version 2.0
작성자 : 문태준 문태준

- 문서소개
- 문서변경내역
- 관련자료
- LDAP란 무엇인가?
  - 디렉토리 서비스와 LDAP
  - LDAP 들여다 보기
- 공개형 LDAP엔진 OpenLDAP
  - OpenLDAP에 대한 간단한 소개
  - LDAPv2 와 LDAPv3의 차이점
  - openldap 설치
  - 기본설정설명
  - 로그레벨 및 로그조정
  - 기본적인 구동
- 좀더 자세히 slapd.conf를 살펴보자
  - ACL(Access Control List)
  - 패스워드 인증방식 설명
  - Indexing의 설명
- 기본적인 OpenLDAP의 콘솔명령 사용
  - ldapadd
  - ldapsearch
  - ldapmodify
  - ldapdelete
  - ldapmodrdn
- LDIF(LDAP Data Interchange Format)
  - 엔트리 내용을 보여주는 LDIF
  - 엔트리 변경에 관련한 LDIF
- LDAP 스키마의 작성
  - openldap 에서 기본 배포되는 스키마
  - OpenLDAP v1.2.x에서의 스키마 작성
  - OpenLDAP v 2.0.x에서의 스키마 작성
- 보안
  - 네트워크 보안
  - 데이터 무결성, 기밀성 보호
  - 인증 방법
- 분산 디렉토리 서비스 구현
  - Subordinate Knowledge Information
  - Superior Knowledge Information
- 리플리케이션
- OpenLDAP의 관리
  - BackUp(백업)
  - 퍼포먼스를 늘리기위한 방법들.
  - LDAP GUI 관리툴로 쉽게 관리할 수는 없을까?
- FAQ(Frequently Asked Questions)
- PHP 샘플예제
  - 관련자료
  - LDAP 프로그래밍 작동 방식
  - 검색하기
  - 입력/수정하기
  - 삭제하기
  - SSL/TLS 사용하여 php프로그래밍하기

문서소개

- 이 문서는 http://database.sarang.net 에 있는 "LDAP의 모든것 ver 20011126" (박근오님 작성) 문서를 기초로 하여 일부 수정하였습니다. 문서를 작성해주신 박근오님에게 감사의 말을 전합니다. 또한 openldap 문서와 오렐리의 ldap 서적을 일부 참고하여 보강하였습니다.
- 원자료 URL : http://database.sarang.net/?criteria=ldap&subcrit=tutorials Tutorials
- 위의 원자료에서 일부 변경된 내용을 반영하였음
- 예제는 Linux CenOS 4.4 에서 2.2.x ldap rpm 설치를 이용하였음

문서변경내역

- 2007.5.23 SSL/TLS 부분 틀린 내용 수정(SSL/TLS 사용하여 php 프로그래밍 하기 부분)
- 2007.3.19 SSL/TLS PHP 예제코드 추가

관련자료

- LDAP에 대한 한글자료는 DSN의 자료 1개와 KLDP의 LDAP 하우투 및 기타 몇개의 문서가 있습니다. 상세한 내용은 영문자료를 보아야 합니다.
- http://database.sarang.net/?inc=read&aid=1243&criteria=ldap&subcrit=tutorials&id=&limit=20&keyword=&page=1 : LDAP의 모든것 ver 20011126. DSN에 2001년 올라왔던 ldap 전반적인 자료. 국내에 ldap 에 대한 한글자료가 별로 없는데 그나마 상세하게 ldap 에 대한 설명이 들은 한글문서입니다. 전체적으로는 LDAP기초부터 기본적인 사용법을 담고 있어 처음에 참고를 할 만 합니다.
- http://wiki.kldp.org/wiki.php/LinuxdocSgml/LDAP-HOWTO : KLDP LDAP 하우투자료
- O'REILLY 의 LDAP System Administration 서적 : LDAP 전반적인 설명을 담고 있으며 각종 애플리케이션을 ldap으로 통합하는 경우에 대한 상세한 자료를 제공하고 있음
- http://www.openldap.org/doc/admin23/ openldap 문서 : openldap에 대한 기본 사용법은 openldap 에서 제공하는 문서를 참고
- LDAP을 이용한 계정통합, 각종 애플리케이션 연동 : 문태준 작성. LDAP 의 상세 활용에 대한 정보가 들어있음. 개인홈피 또는 KLDP 또는 DSN에서 찾을 수 있음. 현재 문서에서 빠진 TLS, 리플리케이션 및 각종 세부 자료가 있음

LDAP란 무엇인가?

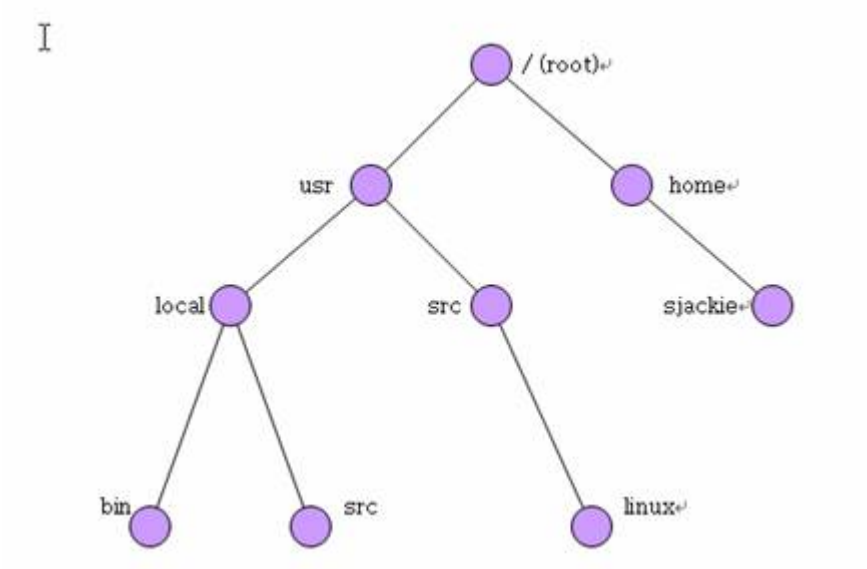
디렉토리 서비스와 LDAP

LDAP란 무엇일까? Lightweight Directory Access Protocol이라는 말인데 우리말로 하면 '경량의 디렉토리 액세스 프로토콜'이라는 말이 된다. 그럼 디렉토리란 무엇일까? 디렉토리란 특별한 형태의 데이터베이스라고 할 수가 있다. 그리고 쓰기 작업보다 읽기 작업이 더 많을 뿐 아니라 어떤 것을 찾는 작업이 많은 곳에 더더욱 적합한 서비스라고 할 수가 있다. 현재로부터 시간을 조금 거슬러 올라가서 1980년대 말에 특정분야의 디렉토리 서비스의 이용,개발 요구가 높아감에 따라 CCITT(International Telegraph and Telephone Consultative Committee, 현재는 ITU이다)와 ISO(International Organization for Standardization) 두 단체가 함께 X.500이라는 디렉토리 서비스 표준을 만들기 시작하였다. 결국 1990년에 CCITT가 표준을 발표했고 1993년,1997년 몇번의 수정작업을 거쳐 현재에 이르렀다. 이 X.500은 최초의 일반적인 목적의 디렉토리 시스템이었고 다양한 쿼리를 사용하는 강력한 검색기능을 제공하였을 뿐만 아니라 서버와 데이터의 분산이 용이했고 그리고 무엇보다도 특정 운영체제나 특정 네트웍,특정 응용프로그램에 구애받지 않고 사용될 수 있는 표준이라는 점이 눈길을 끌 수 있었다.

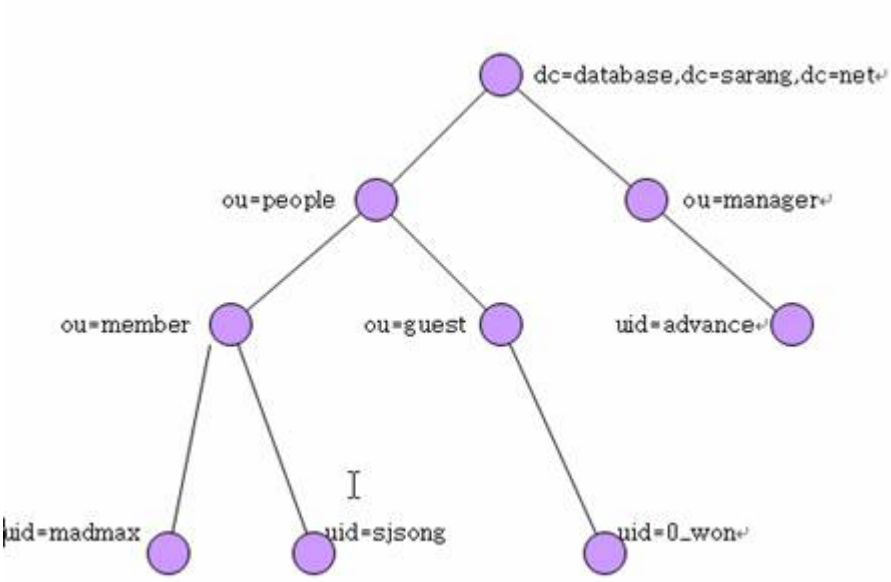
하지만 X.500 개발자들은 DAP(X.500의 directory client access protocol)가 너무 방대한데다 복잡하고 구현하기 어렵다는 점 때문에 그당시의 일반 PC급에서는 적용해서 사용하기가 힘들다는걸 알았고 이의 해결책을 모색하기 시작했고 그렇게 해서 나온 것이 LDAP이다. LDAP는 DAP의 기능을 거의 다 지원을 했고 복잡했던 부분이나 잘 쓰이지 않았던 부분은 단순화하거나 없애버렸다. 그리고 대부분의 데이터 형식에 있어서 단순한 문자열을 사용하므로써 구현을 단순화하고 퍼포먼스를 늘릴수가 있었다. 이렇게 LDAP는 처음에 X.500 디렉토리 서비스의 프론트엔드로 사용되었다. 그후 최초이면서 많이 알려진 미시건대학의 LDAP(U-M LDAP)가 나오게되었고 현재 많은 상용 또는 오픈소스의 LDAP제품들이 나와있다.

LDAP 들여다 보기

이제 본격적으로 LDAP를 들여다 볼 시간이 된 것 같다. 아래의 두개의 그림을 보자.



[~joon:그림] 파일 시스템에서의 간단한 디렉토리의 구조의 예



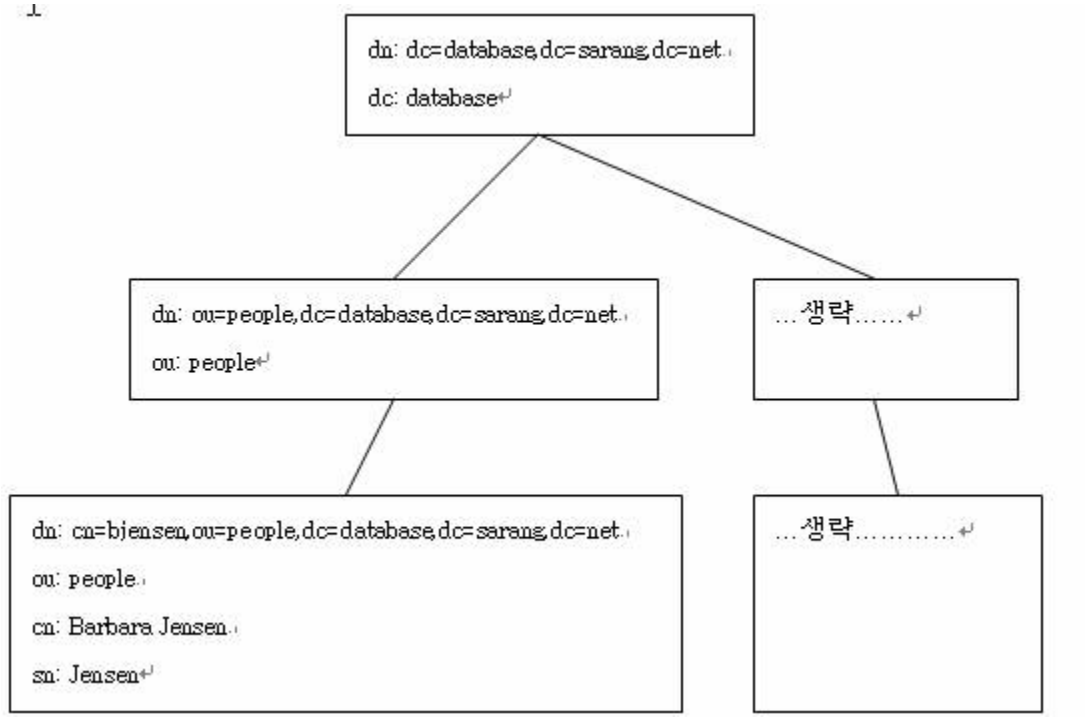
[~joon:그림] LDAP에서의 간단한 디렉토리의 구조의 예

위 두 그림들은 파일 시스템의 구조와 LDAP 디렉토리 구조의 대표적인 예를 보여준다. 이러한 LDAP 디렉토리 트리 구조를 특별히 DIT(Directory Information Tree)라고 부른다. LDAP 트리(Tree) 구조에서 각 노드들을 엔트리(Entry)라고 부르고 엔트리는 LDAP에서 하나의 데이터를 나타낸다. RDBMS와 비교를 한다면 하나의 레코드와 일치한다고 할수 있다. 모든 엔트리는 그 자신의 위치와 고유성을 나타내는 DN(Distinguished Name)으로 구분된다. 이는 마치 파일 시스템에서의 디렉토리 구조를 보면서 "최상위에 루트디렉토리가 있고 그아래에 /usr 디렉토리와 /home 디렉토리가 있고 /usr 디렉토리 아래에는 /usr/local 과 /usr/src 두개의 디렉토리가 있고 등등등..."으로 말할 수가 있는 것 처럼 위의 LDAP 디렉토리 구조 그림에서 트리 구조의 최상위 DN은 dc=database,dc=sarang,dc=net 이고 이 엔트리의 아래에 두개의 엔트리가 존재한다. 이 두개의 엔트리의 DN은 각각 ou=people, dc=database, dc=sarang, dc=net 과 ou=manager, dc=database, dc=sarang, dc=net 이다. 이것은 우리가 파일시스템을 읽을 때 하위 디렉토리로 내려갈수록 /usr/local/bin과 같이 오른쪽으로 붙여나가는 방식과는 반대로 하위 엔트리로 내려갈수록 왼쪽으로 붙여나가는 방식이다. 그리고 위 예제의 구조를 보면 "데이터베이스 사랑넷에 people(사람들) 그룹과 manager(관리자) 그룹으로 나뉘고 관리자 그룹에는 유저 아이디가 advance라는 관리자가 한명 있으며 사람들 그룹아래로는 member(회원)그룹과 guest(손님)그룹이 존재한다." 라는 식으로 이해를 할수 있겠다. 이제 엔트리,DN 이라는 것에

대해 이해를 했을것이다.
위의 DIT 그림에서 각각의 엔트리의 위치를 나타내는 전체 DN이 아니라 단지 ou=people과 같은 앞부분만 써놓았는데 이것을 Relative Distinguished Name. 즉, RDN이라고 한다. 파일 시스템 디렉토리 구조에서 절대 패스, 상대 패스가 있는것과 같이 LDAP도 DN,RDN으로 (굳이 우리말로 하면 절대 DN, 상대 DN정도로 표현하면 될까?) 나타낼수 있다.
이제 엔트리에 대해서 좀더 자세히 보기로 하자.

Attribute type	Attribute value
cn	Barbara Jensen Babs Jensen
sn	Jensen
telephonenumber	+1 408 555 1212
mail	babs@foo.bar.com

위의 표는 하나의 엔트리를 나타낸 예이다. 하나의 엔트리는 마치 RDBMS의 필드처럼 attribute(속성)들을 가지고 이러한 attribute의 값들은 RDBMS의 하나의 필드값과는 다르게 하나 이상의 값을 가질수가 있다. 위이 예에서 cn이란 common name을,sn은 sirname을 각각 대표한다. organization unit을 ou라고 쓰는 것 처럼 왜 그러면 이렇게 줄여서 사용하는 것일까? 이유는 ou(organization unit), o(organization), c(country), cn(common name)과 같이 dn에 쓰이는 attribute들은 보다 간결한 dn을 위해서 축약이 될 필요가 있어서 이다.
한가지 설명에서 빠진 것이 있는데 바로 objectclass이다. 모든 엔트리는 한가지 이상의 obejctclass에 속하게 되며 objectclass의 정의대로 attribute를 가지게 된다. 쉽게 비유를 하자면 objectclass란 붕어빵을 찍어내는 붕어빵 틀과 같다고 할 수가 있겠다. 붕어빵틀에 생긴대로 붕어빵이 나오듯이 우리가 objectclass를 정의한대로 엔트리가 생성되는 것이다. 이러한 objectclass에 대해서 좀더 자세한 이해를 하여야 할 필요가 있는데 지금 당장 알고 싶은 분은 'LDAP 스키마 작성'섹션을 지금 읽어보아도 좋다. 다음 그림은 전형적인 LDAP의 구조를 엔트리 내용을 좀더 자세히 해서 보여주는 예이다.



[~joon:그림] 좀더 자세히 그리 엔트리와 LDAP디렉토리 구조

공개형 LDAP엔진 OpenLDAP

OpenLDAP에 대한 간단한 소개

OpenLDAP는 LDAP의 전신이라고 할수 있는 Umich(미시건대학) LDAP 3.3을 기반으로 새로이 만든 LDAP프로젝트의

산물이다. 오픈소스 정책을 따르면서도 상용 LDAP서버 못지 않은 응용프로그램들과 서버툴들을 제공하겠다는 목적아래 인터넷의 많은 개발자들이 자원해서 이 프로젝트를 돕고 있다.
현재 OpenLDAP는 2.3.x 버전의 안정버전을 내놓고 있다. 1.2.x버전의 경우 LDAP v2 표준을 지원하는 반면 2.x 버전은 LDAP v3 표준을 기반으로 만들어졌다. V2와 V3의 차이점을 알고 싶다면 LDAP RFC문서를 참고한다. 그럼 이제 이 OpenLDAP라는 것을 설치해보자.

LDAPv2 와 LDAPv3의 차이점

LDAPv3 was developed in the late 1990's to replace LDAPv2. LDAPv3 adds the following features to LDAP:

- Strong authentication and data security services via SASL
- Certificate authentication and data security services via TLS (SSL)
- Internationalization through the use of Unicode
- Referrals and Continuations
- Schema Discovery
- Extensibility (controls, extended operations, and more)

LDAPv2 is historic (RFC3494). As most so-called LDAPv2 implementations (including slapd(8)) do not conform to the LDAPv2 technical specification, interoperatibility amongst implementations claiming LDAPv2 support is limited. As LDAPv2 differs significantly from LDAPv3, deploying both LDAPv2 and LDAPv3 simultaneously is quite problematic. LDAPv2 should be avoided. LDAPv2 is disabled by default.

openldap 설치

- openldap 설치는 openldap 사이트의 문서를 참고로 한다.
- http://www.openldap.org/doc/admin23/ 관리자가이드 4. Building and Installing OpenLDAP Software
- A Quick-Start Guide : http://www.openldap.org/doc/admin23/quickstart.html
- CentOS 에서는 RPM으로 설치를 한다. CentOS 에서 LDAP RPM은 최신버전인 2.3.X 가 아니라 2.2.x 이다. 2.3에서는 설정파일도 동적으로 관리하는데 2.2에서는 설정파일을 수동으로 편집하고 대몬을 재시작해야 하는 불편이 있다.
- RPM으로 설치한 경우 대몬 시작은 /etc/init.d/ldap start 를 하면 된다.

```
# yum install -y openldap-servers
# rpm -qa | grep openldap
openldap-2.2.13-6.4E
openldap-devel-2.2.13-6.4E
openldap-clients-2.2.13-6.4E
openldap-servers-2.2.13-6.4E
```

- BackEnd DB에 대해서 : 설치를 시작하기 전에 BackEnd DB에 대해서 잠깐 설명을 할 필요를 느낀다. OpenLDAP는 그 자체의 저장구조를 가지고 있지 않고 다양한 저장구조를 붙여서 사용을 할 수가 있다. 버클리DB, GDBM, NDBM, Shell, Passwd, 심지어 SQL BackEnd도 지원을 한다. 다양한 BackEnd DB를 지원하지만 오직 최고의 성능을 내기위해 OpenLDAP측에서 권장하는 BackEnd DB는 버클리DB이다. 현재 CentOS 4.4 에는 db4 rpm이 버클리db에 해당한다.

```
# rpm -qa | grep db4
db4-utils-4.2.52-7.1
db4-4.2.52-7.1
db4-devel-4.2.52-7.1
```

- 환경설정파일은 rpm으로 설치하였을 경우 /etc/openldap/slapd.conf 이다.

기본설정설명

- 설정의 기본 방식

다음은 slapd.conf의 기본 설정 방식을 보여준다.

```
# ### ### ### ##
<global config directives>
### include### ### ACL#### ### ##.

       # first database definition & config directives
       database <typeA>
       <database-specific directives>
       ### dn, ####, ### ####, ##ACL#### ### ##.

       # second database definition & config directives
       database <typeB>
       <database-specific directives>
       ### dn, ####, ### ####, ##ACL#### ### ##.
```

위와 같이 두개이상의 database지시자가 올수가 있어서 한 개의 서버에 다중 데이터베이스를 구축할수도 있다.
Ex) dc=database,dc=sarang,dc=net과 dc=plug,dc=sarang,dc=net을 함께 관리할 수가 있게된다.
위의 간단한 전체 구조를 보았지만 사실 아직 잘 이해가 안가리라고 생각한다. 다음의 대략적인 전체 설정 파일을 보고 이해한 다음 다시 여기로 와서 본다면 무슨뜻인지 감이 잡힐것이다.

- 대략적인 전체 설정 파일
  slapd.conf파일의 내용을 간단히 살펴보자. 여기에는 빠진 옵션들이 많다. rpm으로 설치한 파일의 내용을 살펴보면 되고 여기는 간단한 예제로서만 보기 바란다.

```
   #
   # See slapd.conf(5) for details on configuration options.
   # This file should NOT be world readable.
   #
   include         /usr/local/ldap/etc/openldap/slapd.at.conf
   # objectclass# ## attribute# ### ## ### include##.
   include         /usr/local/ldap/etc/openldap/slapd.oc.conf
   # objectclass# ### ## ### include##.

   schemacheck     off
   # LDAP add,modify## ### # #### ### #### #### ### ##.
   # off# ## ## ### #### ### ##### ### ### # ## #### #### #### ####### ####.

   pidfile         /usr/local/ldap/var/slapd.pid
   # slapd #### id number

   argsfile        /usr/local/ldap/var/slapd.args
   # slapd argument

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

database        bdb

   # suffix#  ldbm### ### Base Dn# #### #### ### suffix# #### ### #### ### #####.
   suffix          "dc=database, dc=sarang, dc=net"

   # rootdn : #### #### ### dn#### ####.
   rootdn          "cn=DsnManager, dc=database, dc=sarang, dc=net"

   # rootpw : ### ##### ####.
   rootpw          dkagh

   #
   # DB #### #### ####
   # ## #### ###### ### #### ## ##### ##### ### mkdir# ##### ## ### ## slapd# ##### ##.
   # ### ## #### ### ##### ## slapd# # # #### #### ## ##### ### ## ###.
   directory       /var/lib/ldap
```

위에서는 간단한 설정만을 엿보았지만 실제로 좀더 자세히 설정을 조정할수 있으므로 slapd.conf의 맨페이지를 꼼꼼히 읽어보기 바란다. (참고로 여기서는 ACL설정이라든지 캐쉬설정을 뺐다. 다음 섹션에서 좀더 자세히 알아보기로 하자.)

rpm 으로 설치한 경우 기본설정을 이용하되 database 지정, 기본 dn (suffix), rootdn, rootpw 를 바꾸어서 사용하면 된다. 세부설정을 하는 경우에는 스키마 확장, 로그레벨 조정, TLS 설정, 인덱스 조정, ACI 조정, replication 설정등을 추가할 수 있다.

로그레벨 및 로그조정

- 로그레벨은 다음과 같다. 256이 기본값으로 세부적인 정보를 보길 원한다면 loglevel을 조정하면 된다.

Level	Description
-1	enable all debugging
0	no debugging
1	trace function calls
2	debug packet handling
4	heavy trace debugging
8	connection management
16	print out packets sent and received
32	search filter processing
64	configuration file processing
128	access control list processing
256	stats log connections/operations/results
512	stats log entries sent
1024	print communication with shell backends
2048	print entry parsing debugging

- syslog 남기기
  LDAP은 리눅스에서 LOG_LOCAL4 facility를 사용하므로 /etc/syslog.conf 에 아래의 설정을 한다. ldap만 별도 파일로 저장할 수도 있다. 이 경우에는 로그로테이션을 주기적으로 해주어야 한다.

```
# grep local4 /etc/syslog.conf
local4.*                                                /var/log/messages
```

이경우 syslogd 를 다시 재시작해주어야 한다.

참고로 openldap 문서에도 로그레벨에 대한 내용은 있지만 남겨진 로그를 어떻게 분석하면 되는지에 대해서는 상세한 설명은 없었다. 이에 대해서는 작동방식은 비슷할 것이라 여겨지므로 레드햇 디렉토리 서버의 매뉴얼을 참고하면 될 듯 하다.
이에 대한 내용은 http://www.redhat.com/docs/manuals/dir-server/^ ^ 레드햇 디렉토리 서버 매뉴얼 중에서 Configuration, Command, and File Reference 의 Chapter 5 Access Log and Connection Code Reference 를 참고한다. 여기서 로그에 남는 기록이 어떤 에러코드인지 설명을 참고하자.

기본적인 구동

- 소스 설치시 구동하기
  slapd의 시작은 OpenLDAP의 버전에 따라 약간은 틀리지만 크게 다를 것은 없다. 아래 예제는 모두 설치경로(configure스크립트 실행시 prefix의 설정에 따른 경로)를 /usr/local/ldap로 설정했을 때를 가정한 것이다.

- 2.x 의 slapd 구동

```
$ cd /usr/local/ldap
      $ ./libexec/slapd -h ldap://localhost:389/ -f etc/openldap/slapd.conf
```

LDAP의 기본 서비스 포트 번호는 389번이다.

- 올바로 데몬이 수행되고 있는지를 확인해 보자.

```
$ pstree|grep slapd
```

이제 기본적인 설치와 설정은 모두 끝이 났다고 할수 있다. 아래에서 조금더 자세히 설정을 알아보도록 하겠지만 혹시 테스트를 먼저 빨리해보고 싶은 분은 '기본적인 OpenLDAP콘솔 명령 사용' 섹션을 먼저 읽어보시기 바란다.

- RPM 설치시는 위에서 설명을 하였다.

```
# /etc/init.d/ldap start
```

좀더 자세히 slapd.conf를 살펴보자

OpenLDAP의 설정파일인 slapd.conf는 의외로 간단한 구조를 가졌으면서도 관리자의 설정 여하에 따라서 매우 복잡한 설정을 가질수 있고 이에 따라 매우 유연하고 강력한 설정을 해줄수도 있게된다. 이러한 slapd.conf의 몇가지 자세히 알아보아야 할 설정면면을 이제부터 살펴보기로 하겠다.

ACL(Access Control List)

ACL이란 마치 우리가 파일시스템에서 파일과 디렉토리에 소유자와 권한을 설정해 주는 것과 같이 각각의 엔트리에 사용자마다의 접근권한을 설정해 주는 일을 한다. 그리고 이러한 ACL설정으로 LDAP내 각각의 entry,attribute의 접근에 대해 권한을 설정할 수가 있으며 접근할수 있는 호스트에 대한 설정도 할 수가 있다. 그러므로 LDAP관리자는 반드시 데이터의 중요도에 따라서 ACL을 설정해야하며 이러한 설정이 올바르게 적용되고 있는지 확인도 해야 할것이다. 일단 ACL자체에 대한 정의(BNF)를 보기전에 머리가 아프지 않도록 간단한 형식을 들여다 보기로 하자.
ACL의 기본형식은 다음과 같다.

```
access to <##### ### #>
        by <####> <### ##>

<access directive> ::= access to <what>
                [by <who> <access> <control>]+
        <what> ::= * | [ dn[.<target style>]=<regex>]
                [filter=<ldapfilter>] [attrs=<attrlist>]
        <target style> ::= regex | base | one | subtree | children
        <attrlist> ::= <attr> | <attr> , <attrlist>
        <attr> ::= <attrname> | entry | children
        <who> ::= [* | anonymous | users | self |
                dn[.<subject style>]=<regex>]
                [dnattr=<attrname> ]
                [group[/<objectclass>[/<attrname>][.<basic style>]]=<regex> ]
                [peername[.<basic style>]=<regex>]
                [sockname[.<basic style>]=<regex>]
                [domain[.<basic style>]=<regex>]
                [sockurl[.<basic style>]=<regex>]
                [set=<setspec>]
                [aci=<attrname>]
        <subject style> ::= regex | exact | base | one | subtree | children
        <basic style> ::= regex | exact
        <access> ::= [self]{<level>|<priv>}
```

```
        <level> ::= none | auth | compare | search | read | write
        <priv> ::= {=|+|-}{w|r|s|c|x}+
        <control> ::= [stop | continue | break]
```

별다른 설정이 없을 경우 기본값은 모든 사용자에게 read 권한이다. 보안을 위해서는 최대한 접근을 제한해야 할 것이다.

what 에서는 접근할 엔트리, 속성을 지정한다. dn, scope (base, one, subtree, children), 필터, 속성(attr)을 이용할 수 있다.

```
to *
        to dn[.<basic-style>]=<regex>
        to dn.<scope-style>=<DN>
```

첫번째는 모든 엔트리에 해당하며 두번째는 정규표현식을 이용하며 세번째는 해당 DN의 요청 범위(scope)를 지정한다. scope 는 base, one, subtree, children이 있다. base 는 해당 DN만 해당하고 one은 엔트리의 부모(parents)가 해당 DN일 경우이며 subtree는 해당 dn에 속하는 모든 엔트리이다. children는 해당 dn 밑에 있는 엔트리로 지정한 dn 자체는 포함하지 않는다. ldapsearch 에서도 이에 대해서 설명을 하고 있다. 말로만 설명하면 헷갈릴 것이며 openldap 문서에서 6장 Access Control을 참고하기 바란다.

who 는 누구에게 권한을 줄 것인가이다.

Specifier	Entities
	익명 사용자와 인증한 사용자를 포함한 전체 사용자
anonymous	익명사용자
users	인증받은 사용자
self	목적 엔트리와 연관된 사용자
dn[.<basic-style>]=<regex>	정규표현식에 연결되는 사용자
dn.<scope-style>=<DN>	해당 DN의 scope 에 해당하는 사용자

access 에서는 부여할 권한을 지정한다.

Level	Privileges	Description
none	=0	접근하지 못함
auth	=x	인증받아야 함.
compare	=cx	compare attribues 에 접근 (전화번호가 374-1160 입니까?)
search	=scx	검색필터를 사용할 수 있는 권한 (전화번호가 374로 시작하는 목록이 몇 개 있습니까?
read	=rscx	검색결과를 볼 수 있는 권한
write	=wrscx	특정 속성을 업데이트할 수 있는 권한

그럼 일반적인 설정 예제를 한번 보자.

```
access to *
        by self write
        by users read
        by anonymous read
```

```
        by * none
```

'access to *' 부분은 모든엔트리(*)에 대한 접근권한을 정의하겠다는 뜻이다. 정규표현식을 써서 'access to dn.subtree="dc=database,dc=sarang,dc=net" '이라고 하면 dc=database,dc=sarang,dc=net을 dn의 한부분으로 가지는 모든 엔트리에 대해 접근권한을 정의하겠다는 뜻이다. 이제 그 아래에서부터 모든 엔트리에 대한 접근권한 정의가 시작된다.
'by self write'란 엔트리자신(self)이 자기자신의 엔트리에 대해 쓰기권한을 가질수 있다는 것이다. 좀 혼란스러운 내용일수가 있는데 만일 cn=홍길동,dc=example,dc=com 이라는 person 엔트리가 LDAP에 저장되어 있다면 이 엔트리 자신은 즉 홍길동이란 사람은 자신의 dn을 제시하고 userPassword를 제시하므로써 자기자신의 정보에 쓰기권한을 가질수가 있다는 말이된다. 그러므로 홍길동이란 사람은 cn=짱구,dc=example,dc=com이라는 엔트리의 내용에는 쓰기 접근을 할 수가 없다. 물론 관리자 dn과 패스워드로 접근한다면 어떤 엔트리에도 쓰기작업이 허용된다.
'by users read' 이말은 무엇일까? dn이 존재하는 즉 dn이 존재하고 패스워드를 제시해서 인증된 사용자에게만 read권한을 주겠다는 뜻이된다. 'by anonymous read' 이것은 dn과 패스워드를 제시하지 않은 사용자 즉 anonymous 사용자에게 read권한을 주겠다는 뜻이다.
'by * none'란 그외의 모든 상황에 대해서는 권한을 주지않겠다는 뜻이다. 특정한 권한을 지정하지 않으면 none이 기본적용된다.

위의 간단한 예제를 통해서 조금은 아 ACL이란 이런것이구나 하는 감이 왔으리라 생각한다. 하지만 그냥 이해만으로는 부족하니 직접한번 설정을 하고 권한설정이 제대로 되었는지 테스트를 해보기를 바란다.

이제 특정 attribute에 대해서 설정을 해보자.

```
access to attr=userpassword
        by self write
        by * none
```

이라고 설정을 한다면 userpassword 속성에 대해서만 접근권한을 정의하게 된다. 그리고 이때 권한을 설정하고자 하는 것이 복수개라면 콤마를 구분자로 다음과 같이 쓴다.
access to attrs=userpassword,uid,sn
attr에서 복수개일 때 attrs로 바뀐것에도 주의를 하기 바란다.

전체적인 예를 하나 보자.

아래에서는 특정 dn의 범위를 정하여 제한을 세부적으로 하고 있다.(dn.subtree)

```
# userPasswd  attr # ### ### ### ##(self write) ##(auth)# ## ### # ### ##.
access to dn.subtree="dc=samjung,dc=com" attr=userPassword
        by self write
        by * auth
# attrs # #### ## attr# ### ### ### # ### ## ## dn# ### ###### ## ### ##.
access to dn.subtree="dc=samjung,dc=com" attrs=mobile,homePhone
        by self write
        by dn.subtree="dc=samjung,dc=com" read
# ou=group,dc=samjung,dc=com # subtree(## dn# #### ## ###)# ####
# cn=linuxadmin,ou=admingroup,dc=samjung,dc=com # ### ### #### # # ### ##.
# # ### ### #### ## ### ## ##### ### # ## ####.
access to dn.subtree="ou=group,dc=samjung,dc=com"
        by
group/groupOfUniqueNames/uniqueMember="cn=linuxadmin,ou=admingroup,dc=samjung,dc=com" write
        by dn.subtree="dc=samjung,dc=com" read

# ### #### #### ### ## ### ### # ### ##. auth # ### ## ## ### # # ## ## ###.
access to *
        by * auth
```

이제 ACL에 대해서 간단히 알아보았다. ACL에 대한 주의점을 마지막으로 설명을 할때가 온 것 같다. 다음의 주의점을 항상 기억하고 있으면 좋을 것이다.

- ° 위의 ACL들은 하나 이상 여러 개의 리스트를 설정하여 쓸수도 있다. 그리고 local.acl.conf같은 파일 이름으로 ACL을 설정하고 slapd.conf에서 include지시자를 사용하여 쓸수도 있다.
  ° 만일 동일한 액세스 항목을 두개 이상 설정하게 되는 실수를 저지른다면 첫번째 설정값만 실행하게 되고 두번째 설정은 적용되지 않게된다.
  ° ACL정의내의 entry,attribute내용을 작성할 때 콤마(,)앞뒤로 공백문자가 있어서는 않된다. dn=" dc=example, dc=com"과 같이 콤마앞뒤로 하나라도 공백문자가 있는 날에는 여러분이 설정한 ACL이 전혀 작동하지 않는수가 있다.(slapd를 띄울 때 아무 에러메시지가 없다고 해서 ACL이 잘 작동하는 것이 아니라는 뜻이다.)
  ° ACL리스트의 순서가 문제가 될때가 있다. 'access to *'와 'access to dn=" dc=example,dc=com"'이 설정되어 있다면 좀더 특정 ACL리스트라고 할수 있는 'access to dn="dc=example,dc=com"'부분이 보다 앞에 위치하여야 한다. 앞의 설정을 포함할 수 있는 좀더 넓은 범위의 ACL을 뒤로 놓는 것이 안전하다.
  ° ACL이 복잡하면 할수록 ldapsearch에 드는 비용은 비싸기 마련이다. 특히 group ACL은 더욱 그렇다. 그러므로 최소한의 필요한 최적의 ACL만을 설정하는것으로 좀더 나은 검색속도를 가질수가 있게된다. 결국 ACL이 간단하면 간단할수록 정규표현식이 적으면 적을수록 퍼포먼스 향상에 많은 도움이 된다.

패스워드 인증방식 설명

OpenLDAP는 RFC 2307 패스워드 방식을 지원한다. 이 방식에는 {CRYPT}, {SHA}, {SSHA}, {MD5}, {SMD5} 등의 방식이 있다. 이러한 패스워드 방식은 slapd.conf내의 rootpw와 attribute인 userPassword에 사용될수 있다. RFC 2307 패스워드 방식에서 {SSHA}방법이 일반적으로 널리 사용되고 crack하기가 어렵다. 반면 {CRYPT}는 특성상 운영체제 의존적인 문제가 있고 (그러므로 항상 slapd가 떠있는 시스템에서 crypt하여 생성해야 하는 점이 있다.) crack되기가 쉬워 그리 권장할만한 방식이 아니다. 이제 이 글의 목적을 벗어나지 않는 범위에서 간단히 알아보고 넘어가겠다.

- plain text
  가장 단순한 방식이며 LDAP데이터의 외부노출에 대해 크게 신경쓰지 않아도 되는 곳이나 보안에 대해 크게 신경을 쓰지 않아도 되는 곳이라면 이 방식을 쓰면된다. 다음은 입력되는 데이터의 예를 보여준다.

```
rootpw  secret       # slapd.conf# ### #### ######.
        userPassword: secret
```

- crypt
  위에서 말한것 처럼 이 방법은 그리 권장하는 방법은 아니지만 손쉽게 쓸수가 있다. 그리고 꼭 주의해야 할점은 이 CRYPT방식은 운영체제 의존적이라 항상 slapd데몬이 떠있는 시스템에서 패스워드 생성을 해야 한다. /etc/passwd(또는 /etc/shadow)에 있는 패스워드를 긁어서 userPassword 또는 rootpw에 '{CRYPT}encrypted_password' 와 같이 입력을 한다. 그런다음 테스트를 해보면 잘 될것이다. 또는 여러분이 직접 스크립트를 작성하여 패스워드를 생성해도 좋다. perl을 사용하여 다음과 같이 해본다.

```
perl -e 'print("{CRYPT}" . crypt("secret","salt") . "n");'
```

  secret는 패스워드를 말하는 것이고 다음 인자는 salt값을 말하는 것이다. 대부분의 시스템에서 잘 작동할 것이지만 다음과 같이 해야 되는 경우도 있을것이다.

```
perl -e 'use Crypt:: PasswdMD5;
        print("{crypt}" . unix_md5_crypt("secret","salt") . "n");'
```

  위의 생성 예는 perl 모듈 Crypt:: PasswdMD5를 필요로 하고 MD5가 설치되어 있어야 한다. 그리고 위의 예가 MD5를 포함하고 있지만
  여전히 {crypt}라는 것을 기억하기 바란다.

- md5, smd5
  다음의 스크립트로 MD5방식의 패스워드를 생성할수 있다.

```
#/usr/bin/perl
        use MIME::Base64;
        use MD5;
        my $passwd = 'secret';
```

```
my $md5 = new MD5;
print "{MD5}" . encode_base64($md5->hash($passwd,''));
```

- sha, ssha
  http://developer.netscape.com/tech/overview/index.html?content=/docs/technote/ldap/pass_sha.html 에서 자세한 도움을 얻을수 있으므로 여기서는 생략하겠다. Perl과 Java로 생성하는 소스와 인증하는 소스를 얻을수 있고 SHA에 관련한 RFC위치,레퍼런스를 얻을수 있다.

- ° OpenLDAP v2.x 부터 slappasswd라는 커맨드라인 명령어가 포함되었다. 그래서 아주 간단히 {crypt}, {md5}, {smd5}, {sha}, {ssha} 패스워드를 만들어낼수가 있다. 사용방법은 다음과 같다.

```
# slappasswd -h {sha}
```

위와 같이 치면 패스워드를 물어보고 두번 확인 입력을 하면 해싱(Hashing)되어 생성된 패스워드를 알려준다. 그냥 아무 인자 없이 slappasswd만 치게 되면 기본값으로 {ssha}가 넘겨진다. 이것을 복사하여 slapd.conf의 rootpw에 갖다 붙이거나 하면된다. 좀더 자세한 사항이 필요하다면 맨페이지를 참고하기 바란다.

참고 : TLS/SSL , Kerberos, SASL
다음의 링크를 따라가면 유용한 정보를 얻을수 있을것이다.

- How do I use TLS/SSL ?
  http://www.openldap.org/faq/data/cache/185.html

- Using TLS/SSL with slapd, a quick guide
  http://www.openldap.org/lists/openldap-devel/199908/msg00039.html

- Using OpenLDAP with SSL/TLS in OpenLDAP 1.2.x
  http://www.openldap.org/faq/data/cache/65.html

Indexing의 설명

최고의 퍼포먼스를 유지하기 위해서 인덱싱이 꼭 필요한 attribute만을 명시하고 자료를 저장하면 된다. 인덱싱을 설정한후 slapd 데몬을 다시 기동시키고 데이타 입력작업을 하면 인덱스가 생성되기 시작한다. 하지만 이미 입력되어 있는 데이타에 대해서는 slapd가 알아서 인덱싱을 해주지 않는다. 그러므로 이미 입력된 데이타에 대해서 따로 인덱싱 작업을 해줄 필요가 있다. 또는 새로이 인덱싱이 필요한 attribute가 생겼을때도 마찬가지로 추가적인 인덱싱 작업을 관리자가 직접 해주어야 한다. 인덱싱은 다음과 같이 slapd.conf파일에 명시한다.

```
index cn,sn,givenname,mail
```

이제 새로이 slapd 데몬을 기동시킨다. 이미 데이타가 있는 경우를 예로 들어 보면 이전까지 mail이라는 attribute에 대해서는 인덱싱이 이루어 지지 않고 있었다. 그러다가 방금 mail attribute를 인덱싱 하도록 설정했다고 생각하자. 그럼 slapindex 프로그램을 이용하면 된다.

```
slapindex -f <slapdconfigfile>
                [-d <debuglevel>] [-n <databasenumber>|-b <suffix>]
```

이제 데이터 디렉토리에 mail.bdb 라는 인덱싱 파일이 생긴것을 볼수가 있을것이다. 속도 또한 인덱스를 쓰기이전과 달리 눈부시게 향상되었을 것이다.

기본적인 OpenLDAP의 콘솔명령 사용

pstree|grep slapd 로 slapd가 떠있는지를 확인했다면 이제 ldap를 가지고 테스트를 해볼 준비가 된 셈이다. 모든 콘솔명령어에는 공통적으로 들어가는 커맨드라인 옵션이 있다. 그 옵션들은 다음과 같다.
-h : 접속할 LDAP호스트를 설정한다. 설정하지 않으면 로컬호스트를 검색한다.
-p : 접속할 LDAP호스트의 포트번호를 명시한다. 설정하지 않을경우의 디폴트값은 389번 포트이다.
-D : 접속자의 bind dn이다. 마치 우리가 RDBMS에 접속할 때 아이디와 비밀번호가 필요한 것처럼 -D에 자신이 알고 있는 엔트리 dn을 제시한다.
-W 또는 -w password : 비밀번호를 제시한다. -W와 -w의 차이점은 -w뒤에는 비밀번호를 적어줘야 하지만 -W뒤에는 비밀번호를 적어주지 않고 프로그램이 오퍼레이션을 시작하기 전에 사용자에게 물어보도록 한다. 해킹당할 우려가 있으므로 되도록 -w보다는 -W를 사용하는 습관을 가지도록 한다.
-x : simple authentication. 기본 인증방식임. 이를 지정하지 않을 경우 ldap에 접속을 하지 못한다.
-d debuglevel : 디버깅을 해야할 필요가 있을 때 필요하다.

ldapadd

이제 처음으로 띄운 LDAP 서버에 데이터를 한번 넣어보도록 하자. 커맨드라인 명령을 사용할경우 두가지 방법으로 데이터 입력이 가능하다.

- 커맨드라인상에서 바로 입력
  커맨드라인 상의 입력은 대개 단순히 하나의 데이터를 입력할 때 편하긴 하지만 거의 쓸일이 없다. 만일 넣었던 데이터와 유사한 데이터를 계속 넣어야 하는 일이 생기면 여간 번거로운 일이 아닐수 없다. 일단 명령도 익힐 겸 한번 입력해보자. 만일 위의 예제에서 했던 것과 똑같이 설정을 했다면 다음과 같이 입력한다.

```
$ ldapadd -x -D "cn=DsnManager,dc=database,dc=sarang,dc=net" -W << EOF
> dn: dc=database,dc=sarang,dc=net
> dc: database
> objectclass: dcobject
> EOF                                  # ### ## EOF# #### ### ###.
Enter LDAP Password: dkagh            # ### ### ####.
adding new entry dc=database,dc=sarang,dc=net
```

위의 작업은 DIT의 최고 상위 엔트리를 입력하는 작업이다. 이제부터 앞으로 입력하는 엔트리는 모두 이 엔트리의 아래에 위치하게 된다. 그리고 dcobject objectclass의 스키마를 보고 요구사항에 틀리지 않도록(1.2.x버전이라면 etc/openldap/slapd.oc.conf 파일에 있고 2.0.x대 버전의 경우 etc/openldap/schema/core.schema 파일에 정의되어 있다.) 해주어야 한다. 그렇지 않을경우 입려과정에서 에러가 난다. 만일 문법에 틀리는 데도 입력이 잘된다면 slapd.conf에 schemacheck 부분이 off로 되어 있지 않나 보기 바란다. (하지만 입력된 데이터의 옳고 그름을 점검하는 과정도 부하라고 할 수가 있으므로 신뢰할만한 관리자들만이 입력,수정을 할 경우에는 off로 해놓는 것이 더 나을수 있다. 그리고 1.2.x의 slapd.conf의 schemacheck은 디폴트가 off이지만 2.x의 schemacheck은 디폴트가 on이라는 점도 참고할 필요가 있다.)

- LDIF파일 포맷으로 입력

다음 섹션에서 좀더 자세히 설명하겠지만 LDIF(LDAP Data Interchange Format)란 LDAP의 데이터를 텍스트형식으로 표현하는 데이터 포맷이다. LDIF포맷의 파일로 입력하면 똑 같은 타이핑 작업을 피할수 있을뿐 아니라 백업된 데이터의 복구에도 이 방식이 쓰일수 있으므로 유용하다고 할수 있다. Vi와 같은 편집기를 열어 위의 내용을 dsnroot.ldif와 같은 파일이름으로 저장을 한다.

```
$ ldapadd -D "cn=DsnManager,dc=database,dc=sarang,dc=net" -W -f dsnroot.ldif
Enter LDAP Password: dkagh
adding new entry dc=database,dc=sarang,dc=net
```

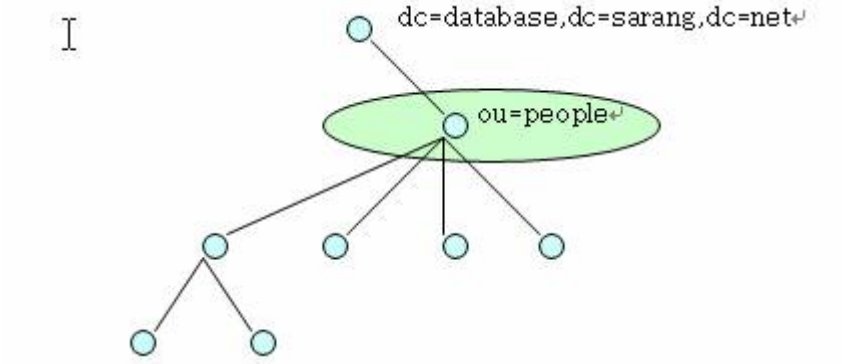
입력이 잘되었다. 우리가 입력한 데이터가 잘 입력이 되었는지 확인작업으로 넘어가기 전에 이 LDIF로 입력할때의 주의사항이 한가지 있다. 흔히 실수할수 있는 일인데 문제는 항상 LDIF포맷 파일에 입력할 엔트리가 두개이상일 경우에 발생한다. 두개 이상의 엔트리일경우 엔트리간의 간격은 빈줄 한줄로 꼭 유지하여야 한다. 꼭 한줄이어야 한다.

두줄이상이 되어버리면 첫 엔트리만이 입력이 되고 두번째 엔트리 데이터부터는 입력이 안된다. 이 사실을 모르고 왜 안될까 계속 고민하는 일은 없기를 바란다. 그리고 입력되는 엔트리 순서에도 신경을 써주어야 한다. 예를 들어 만일 "cn=주소록,dc=database,dc=sarang,dc=net" 과 같은 dn을 가지는 엔트리가 "cn=정재익,cn=주소록,dc=database,dc=sarang,dc=net"과 같은 엔트리보다 아래에 있다면 ldapadd 명령은 탑다운 방식으로 읽어내려가며 입력을 하게 되므로 에러를 내게 된다.
이제 우리가 입력한 데이터가 잘 있는지 확인을 해보자.

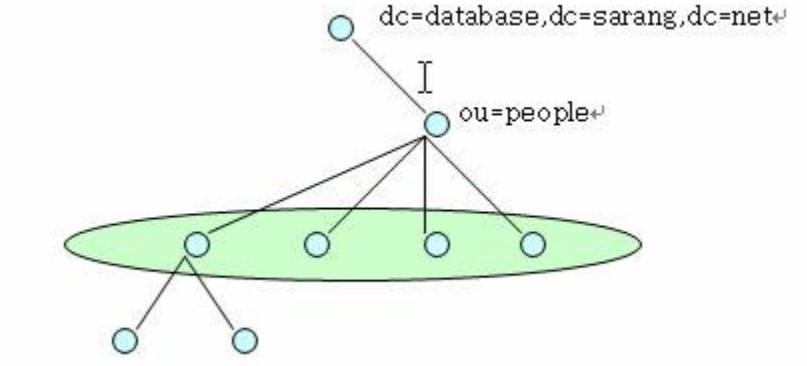
ldapsearch

LDAP를 검색할 때 우리가 SQL을 사용하여 일반적으로 관계형 데이터베이스를 검색하는 것과 같이 검색조건을 설정하고 검색을 하는데, 다음과 같은 네가지 주목할만한 검색조건에 대해서 말을 해보자. (여기서 언급하지 않은 사항들에 대해서는 ldapsearch의 맨페이지를 참조하면 될것이다. 그리고 이 내용은 ldapseach명령에만 해당되는 것이 아니라 LDAP의 검색에 관련된 모든 함수들이나 명령들에도 이 사항들은 공통적으로 해당이 된다.)

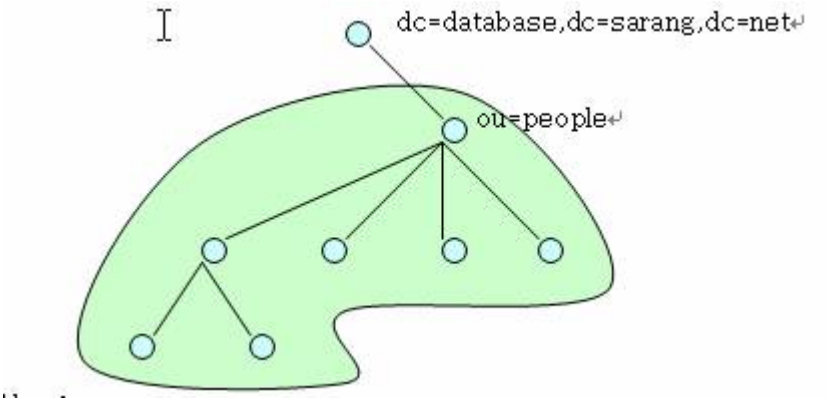
- search scope
  LDAP의 검색 영역을 지정하는 것으로 아래의 세개의 그림으로 충분히 이해가 가리라 생각이 된다. 그리고 base dn 이라는 말을 알아야 하는데 검색을 하는 기준점을 뜻하는 말이다. 무조건 base dn이하 엔트리들에서 검색을 하는데 그 종류가 아래와 같이 세가지로 나뉜다는 것이다. base란 검색하는 base dn으로 제시한 그 엔트리 자체만을 검색한다는 뜻이다. onelevel이란 base dn아래 한단계 하위 엔트리들까지에서만 검색을 한다는 것이다. subtree란 base dn아래의 모든 엔트리에서 검색을 한다는 뜻이다.



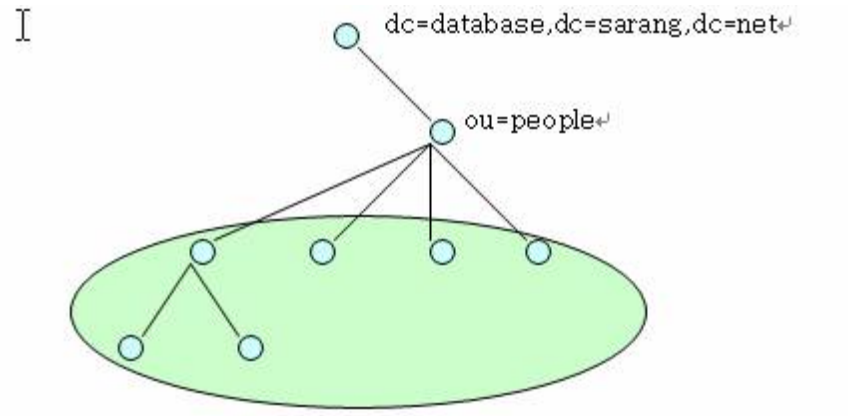
₩~joon:그림 base scope



₩~joon:그림 onelevel scope



₩~joon:그림 subtree scope



₩~joon:그림 children scope

- search filter
  LDAP의 다양한 검색 필터에 대해서 알아보자. 대충 여덟가지 정도의 검색 필터를 사용할 수가 있다.

- equality : '(cn=홍길동)' 과 같이 완전 매칭이 이루어지는 검색 필터를 뜻한다.
- substring : '(cn=길동)' '(cn=홍길)' '(cn=길)' 과 같이 부분 문자열이 같은 것을 검색하는 필터이다.
- approximate : 말뜻 그대로 유사한 단어를 검색하는데 쓰이는데 한글은 되지 않고 영어의 경우 'pipe'를 검색하면 piping 이라든지 유사단어를 검색해준다.
- less than,greater than : '(cn>=kikibon)' 단어순서상의 더 크고 더 작은 조건을 만족하는 검색 결과를 얻을수 있게해주는 검색필터이다.
- presence : '(cn=*)' 와 같이 cn attribute가 있고 없고를 따지는 검색필터이다.
- and : (&(cn=구본규)(telephoneNumber=657)) 과 같이 필터간의 AND 조건 검색을 만들어 준다.
- or : (|(cn=성재철)(telephoneNumber=019*)) 와 같이 필터간의 OR조건 검색을 만들어 준다.
- not : (!(cn=박*)) 과 같이 cn이 '박'으로 시작하지 않는 결과를 돌려받는데 쓰이는 검색 필터이다.

이제 이러한 필터들을 조합해서 좀더 복잡한 조건을 가지는 필터들을 만들어서 검색할수도 있다.

```
Ex) (&(|(cn=#*)(cn=#*))(telephoneNumber=016*))
```

이름이 박 또는 송으로 시작하면서 전화번호는 016으로 시작하는 사람을 찾는 검색 필터 예제이다. 이처럼 검색 필터는 원하는 만큼 겹쳐쓸수가 있지만 복잡하면 복잡할수록 검색에 걸리는 시간은 더 늘어날 것이다.

- time limit
  검색하는데 걸리는 시간에 대해서 제한을 걸어놓는다. 이것은 slapd.conf의 timelimit의 시간제한 범위내에서만 설정가능하다. slapd.conf의 timelimit 보다 더 오랜 시간을 설정할 수는 없다.

- size limit
  검색결과로 돌려받는 결과 데이터 크기에 제한을 한다. 이것도 위와 마찬가지로 slapd.conf의 sizelimit보다 큰 크기로 설정할 수는 없다.

ldapmodify

LDAP서버의 특정 엔트리에 수정을 하고 싶을 때 사용하는 명령이다. 다음 섹션의 'LDIF'에서 '엔트리 변경에 관련한 LDIF'부분을 읽어보면 LDIF로 이 명령을 사용하는 방법에 대해서 설명해 놓았다. 단지 LDIF를 쓰지 않고 수정하는 법은 ldapadd와 유사하다.

```
$ ldapmodify -x -D "cn=DsnManager, dc=database, dc=sarang, dc=net" -W << EOF
> dn: cn=sjackie, ou=people, dc=database,dc=sarang,dc=net
> changetype: modify
> replace: cn
> cn: ##
> -
> add: sn
> sn: #
> -
> delete: jpegPhoto
> EOF
Enter LDAP Password: dkagh
modifying entry "cn=sjackie, ou=people, dc=database, dc=sarang, dc=net"
```

ldapdelete

LDAP서버의 특정 엔트리를 지우고 싶을 때 이 ldapdelete명령을 사용하여 지운다. 쓰는 방법은 다음과 같다.

```
ldapdelete -x -v -D binddn -w passwd dn
ldapdelete -x -v -D binddn -W dn

binddn = "cn=DsnManager, dc=database, dc=sarang, dc=net"
dn = "### ## dn" (leaf node# ### ####.)
```

```
#) $ ldapdelete -x -v -D 'cn=DsnManager, dc=database, dc=sarang, dc=net' -W 'cn=wannadelete,
dc=database, dc=sarang, dc=net'
```

좀더 자세한 옵션들은 맨페이지를 참조하기 바란다.

위의 예제처럼 ldapdelete로는 항상 leaf node만이 삭제 가능하다. 하지만 leaf node가 아닌경우에 서브트리전체를 지우고 싶을 때도 있을것이다. OpenLDAP 2.x의 경우에 -r(recursive)옵션을 주므로써 간단히 주어진 dn이하의 서브트리를 모두 삭제할 수가 있다.

ldapmodrdn

엔트리의 dn과 rdn을 수정하고 싶을때에 사용한다. 만일 다음과 같이 해보자.
수정하기 이전의 엔트리 내용은 다음과 같다.

```
dn: cn=someone,dc=database,dc=sarang,dc=net
objectclass: person
cn: someone
sn: dontknow
```

이 엔트리의 dn과 rdn(cn=someone)을 수정해보기 위해서 다음과 같이 해본다.

```
ldapmodrdn -x -D "cn=DsnManager, dc=database, dc=sarang, dc=net" -W "cn=someone, dc=database,
dc=sarang, dc=net" "cn=kikibon"
```

이제 어떻게 변했는지 알아보자.

```
dn: cn=kikibon,dc=database,dc=sarang,dc=net
objectclass: person
cn: someone
cn: kikibon
sn: dontknow
```

위와 같이 바뀌었을 것이다. 이때 cn: someone 데이터까지 없애버리려고 한다면 ldapmodrdn에 -r 옵션을 주면 된다. 만일 파일로 저장한다음 -f 옵션으로 할 필요가 있다면 위의 두 바뀌어야 할 dn과 새로운 rdn을 차례로 적고 파일로 저장한후 실행하면 될것이다.

LDIF(LDAP Data Interchange Format)

LDAP의 엔트리 데이터를 우리가 읽을수 있고 직접 작성 가능한 텍스트형식으로 보여주는 것이 LDIF이다. 이러한 텍스트라는 점 때문에 두개의 서로다른 BackEnd 데이터베이스를 사용하는 LDAP서버의 데이터들도 LDIF형식으로 export를 하면 서로다른 LDAP서버에 import가 가능하여 서로 다른 LDAP서버들 간의 상호 데이터 호환성을 유지할 수가 있다. 이렇게 되면 LDAP서버를 바꾸는 상황이 와도 그동안 이전 서버에서 사용하였던 데이터에 아무런 수정작업이 없이도 그냥 쓸수가 있게된다. 즉 서로다른 기종의 서로다른 BackEnd데이터 베이스를 사용하는 서로다른 LDAP서버간에 데이터교환이 별 걱정없이 가능하다는 뜻이다. 이러한 LDIF의 형식에는 두가지가 있는데 하나는 디렉토리 엔트리 데이터를 표현하는 형식이고 또한가지는 데이터의 수정,삭제등을 요구하는 데이터 변경에 관련한 형식이 있다. 그럼 이 두가지에 대해서 이제 알아보기로 하자.

엔트리 내용을 보여주는 LDIF

아래의 LDIF형식을 자세히 보자.

```
dn: cn=keuno,dc=database,dc=sarang,dc=net
objectclass: person
sn: park
cn: keuno
cn: sahara
userpassword: mypasswd
description: hi.. If you want to fold a line,you must insert
 a newline character and a space character into the value.
 because some editors do not handle extremely long lines,
 you can fold the line freely.
```

대표적인 LDIF형식의 예를 보여준다. 예를 보듯이 모든 attribute의 값들은 attribute이름과 콜론으로 구분되면서 콜론 다음에 실질적인 값들이 나온다. 그리고 예에서 cn과 같이 두개이상의 값을 가진 attribute들은 각각의 값을 한라인씩 쓴다.(위에서도 말했지만 dn을 제외한 모든 attribute들은 ~~엄밀히 말하면 dn은 attribute가 아니다.~~ 두개이상의 값을 가질수 있다.) 또한가지 주목할 점은 attribute값을 여러줄에 나누어 쓴 description 부분이다. 위와 같이 newline문자와 공백을 값에 넣어주므로써 데이터를 여러줄에 나누어 쓸수가 있다. 하지만 텍스트 에디터가 하나의 긴 라인을 지원한다면 한줄에 써도 무방하다. 자 또 한가지 예를 보기로 하자.

```
dn: cn=someone,dc=database,dc=sarang,dc=net
objectclass: image
cn: someone
```

```
jpegPhoto:: /9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAsICAoIBwsKCQoNDAsNERwSEQ8PESIZG
 hQcKSQrKigkJyctMkA3LTA9MCcnOEw5PUNFSElIKzZPVU5GVEBHSEX/2wBDAQwNDREPESESEiFFL
 icuRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUVFRUX/wAARC
 x5w5BjK49x7VzZFFFDAcqhiRnHBNATNFFIYw/epRRRQIXFGKKKBi4pCKKKBABRRRSA//9k=
```

위의 예를 보듯이 일반적인 아스키 문자로 표현할수 없는 값을 가진 attribute에 대해서는 이것을 프린트 가능한 문자로 바꾸기 위해서 Base64인코딩을 한다. 위의 예에서 jpegPhoto부분을 주목해서 보기 바란다. 이때 구분자는 콜론이 하나가 아니라 두개로 구분을 하고 있다. (참고로 이미지 데이터를 ldif로 변경할때는 base64인코딩을 직접해도 되지만 간단하게 'ldif -b jpegPhoto < image.jpg > image.ldif' 와 같이 사용해도 편하다.)
여기 또다른 LDIF예제가 있다.

```
dn: cn=sahara4,dc=database,dc=sarang,dc=net
objectclass: inetOrgPerson
sn: park
cn: keun-o
jpegPhoto:< file:///home/lastnite/a.jpeg
```

자세히 보면 위의 jpegPhoto부분이 다르다. 이 방식은 OpenLDAP 2.x버전에서만 가능하며 1.2.x버전에서는 지원되지 않는 방식이다.

엔트리 변경에 관련한 LDIF

LDIF는 엔트리 데이터의 표현뿐아니라 데이터의 변경,수정에 관련한 정보를 포함할 수도 있다. 다음의 예를 보자.

```
dn: cn=lee woon-uk,dc=database,dc=sarang,dc=net
changetype: add
objectclass: person
sn: lee
cn: woon-uk
```

주의해서 보아야 할 부분은 changetype이라는 부분이다. 여기에 add를 쓰느냐 delete를 쓰느냐 modify를 쓰느냐에 따라서 작업이 달라진다. 위는 해당 엔트리 데이터를 add시켜달라는 뜻이된다. 직접 테스트를 해보도록 하자.

```
$ ldapmodify -x -D "cn=DsnManager,dc=database,dc=sarang,dc=net" -W -f leeadd.ldif
```

입력이 잘 이루어 질것이다. 이제 삭제를 해보도록 하자.

```
dn: cn=lee woon-uk,dc=database,dc=sarang,dc=net
changetype: delete
```

changetype이 delete로 바뀐것뿐이다. 지우려는 dn을 쓰고 changetype을 delete로 하면 해당 엔트리가 삭제된다. 그럼 변경을 하는 예를 보자. 약간 복잡해 보인다. 테스트는 위의 ldapmodify를 똑같이 쓰면 된다.

```
dn: cn=lee woon-uk,dc=database,dc=sarang,dc=net
changetype: modify
replace: cn
cn: cura
-
add: sn
sn: woonuk
-
delete: description
-
```

이제 changetype이 modify로 바뀌었다. replace는 cn항목을 변경하겠다는 뜻이고 add는 새로운 attribute를 추가하겠다는 뜻이다. delete는 description attribute를 삭제하겠다는 뜻이다. 그리고 '-'문자에 대해서도 눈여겨 볼 필요가 있다.

LDAP 스키마의 작성

기본으로 제공되는 slapd.oc.conf,slapd.at.conf(이상 OpenLDAP 1.2.x)와 schema디렉토리이하에 있는 파일들(이상 OpenLDAP 2.x)의 ObjectClass들로 그대로 사용한다면 편하겠지만 대부분 그렇지 않다. 자신이 원하는 용도에 꼭 맞는 스키마를 작성하므로써 최적화된 LDAP서버를 돌려야 하는 것은 기본이다.

openldap 에서 기본 배포되는 스키마

etc/openldap/schema 디렉토리에 기본 배포되는 스키마 파일이 있다.
core.schema : OpenLDAP core (required)
cosine.schema : Cosine and Internet X.500 (useful)
inetorgperson.schema : InetOrgPerson (useful)
misc.schema : Assorted (experimental)
nis.schema : Network Information Services (FYI)
openldap.schema : OpenLDAP Project (experimental)

OpenLDAP v1.2.x에서의 스키마 작성

일단 안정버전인 OpenLDAP 1.2.x의 slapd.oc.conf의 간단한 objectclass 정의를 보자.

```
objectclass person
        requires
                objectClass,
                sn,
                cn
        allows
                description,
                seeAlso,
                telephoneNumber,
                userPassword
```

objectclass의 정의를 엔트리 데이터 입력시 꼭 있어야 하는(requires) attribute와 있어도 되고 없어도 되는(allows) attribute로 구분하여 정의하고 있다. 상당히 직관적으로 이해하기가 쉽다. Requires,allows는 RDBMS에서의 필드정의시 null,not null을 정의하는 것과 유사해 보이기도 할것이다. 물론 위의 attribute들은 slapd.at.conf에도 정의되어 있는 것이다. 다음은 attribute의 정의이다. slapd.at.conf내용의 일부이다.

```
attribute      jpegphoto      bin
attribute      userpassword  ces
attribute      telephonenumber  tel
attribute      seealso        dn
```

정의 형식은 다음과 같이

```
attribute      ##### attribute##    #####
```

데이터형식에는 bin,cis,ces,tel,dn등이 있고 bin은 binary,cis는 case inexact sensitive, ces는 case exact sensitive, tel은 전화번호 형식, dn은 dn형식이라는 뜻이다. 특이할 만한 사항은 cis인 attribute의 경우는 일부러 정의하지 않아도

된다. Attribute로 정의되지 않았으면서 objectclass의 정의에 있는 attribute에 대해서는 이 cis룰을 적용한다.
그럼 이제 이것을 바탕으로 우리가 필요로 하는 objectclass를 정의해 보자. 위에서 objectclass를 붕어빵틀에 비유를 했었는데 입맛에 맞는 붕어빵틀을 만들어보기로 하자. 가장 간단한 예로 유저 그룹을 생성하는 예를 들어보면 위의 person클래스와 거의 유사하겠지만 추가적으로 유저그룹 사용자의 사진을 가지고 있어야 한다고 생각한다면 allows항목에 jpegPhoto를 추가해 줄수가 있겠다. 한가지 주의할점은 LDAP v3를 따르는 OpenLDAP 2.x의 경우 objectclass간의 상속(inheritance)이 가능한데 LDAP v2를 따르는 OpenLDAP 1.2.x는 이런 상속기능이 없다. 그러므로 상관없이 그냥 하나의 objectclass를 정의해주면 된다. 부산 리눅스 유저 그룹(PLUG)을 예로 든다면 다음과 같이 만들수 있겠다.

```
objectclass PLUGUser
        requires
                objectClass,
                sn,
                cn,
        allows
                email,
                telephone,
                jpegPhoto,
                signature
```

만일 자신이 작성한 objectclass의 정의에 cis속성이 아닌 새로운 attribute가 정의되어 있다면 꼭 attribute정의를 따로 해야한다. 그리고 주의할점은 slapd.oc.conf와 slapd.at.conf파일을 수정하지 말라는 것이다. 자신이 추가한 클래스를 include 하려면 slapd.conf에 objectclass정의와 attribute정의를 쓰던지 아니면 local.oc.conf , local.at.conf와 같은 파일을 만들어서 slapd.conf에서 include 지시자를 써서 포함시키면 된다. 이제까지 OpenLDAP 1.2.x버전에서의 스키마 작성에 대해서 알아보았다.

OpenLDAP v 2.0.x에서의 스키마 작성

이제 LDAP v3를 지원하는 OpenLDAP 2.x에서의 스키마 작성에 대해서 알아보자. 이전버전과는 달리 2.x에서는 LDAP v3표준을 따르면서 많이 달라진 모습을 보인다. etc/openldap/schema 디렉토리 아래의 스키마파일들을 들여다 보자. 기본적으로 포함되는 파일은 core.schema파일이다. 위와 비교가 될수 있도록 person클래스를 여기에 옮겨본다.

```
objectclass ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL
        MUST ( sn $ cn )
        MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )
```

뭔가 좀 많이 다른 것 같이 느껴진다. 일단 이해를 할수 있겠다 싶은 부분은 MUST(requires) 와 MAY(allows)일것이다. 그리고 '$'문자로 attribute들간에 구분을 지어놓았다는 것을 알수 있다. SUP란 top이라는 objectclass의 서브 클래스라는 뜻이다. 상속의 개념이라 생각하면된다.

```
attributeType ( 2.5.4.41 NAME 'name'
                EQUALITY caseIgnoreMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
attributeType ( 2.5.4.3 NAME
                ( 'cn' $ 'commonName' ) SUP name )
```

위에서 보듯이 attribute도 상속의 개념이 있다. SUP name 부분을 보면 attribute 'cn'또는 'commonName'은 위에서 선언한 'name' attribute의 속성을 상속받는다. 그리고 같은 attribute가 두가지 이상의 이름을 가질수도 있다. 'cn'과 'commonName'이 그것이다. 자 LDAP v3의 objectclass,attribute정의는 설명하기 너무 길기 때문에 여기서 생략하고 참고할수 있는 URL을 알려주는 것으로 대신하겠다.
http://www.openldap.org/doc/admin/schema.html
ftp://ftp.isi.edu/in-notes/rfc2252.txt

*Attribute 작성
OpenLDAP 2.0.x에서 새로운 attribute를 작성하기 위해서는 attributetype이라는 지시자를 사용하여 새로운 attribute를 정의한다. 다음은 attributetype지시자를 어떻게 사용할수 있는지를 나타내는 BNF이다. 일단 모두 이해를 할 수는

없더라도 한번 들여다 보기로 하자. attribute는 다음처럼 attributetype지시자와 AttributeTypeDescription을 사용하여 정의된다.

attributetype <RFC2252 AttributeTypeDescription>
위와 같이 attributetype정의는 RFC2252를 따르고 있다. 그러므로 LDAP v3 attribute에 대해서 자세히 알고 싶다면 RFC2252문서를 읽어보도록 한다. 그럼 AttributeTypeDescription 부분이 어떻게 구성 되는지 살펴보자.

```
AttributeTypeDescription = "(" whsp
           numericoid whsp              ; ### attribute# ### OID
         [ "NAME" qdescrs ]             ; ## ### attribute# ##
         [ "DESC" qdstring ]             ; description (attribute# ## ##)
         [ "OBSOLETE" whsp ]          ;
         [ "SUP" woid ]                 ; woid# #### attribute### ### #### ###. ### ##
##(woid)# ### #### ##.
         [ "EQUALITY" woid              ; Matching Rule name
         [ "ORDERING" woid              ; Matching Rule name
         [ "SUBSTR" woid ]               ; Matching Rule name
         [ "SYNTAX" whsp noidlen whsp ] ; attribute# ### ## ####.
         [ "SINGLE-VALUE" whsp ]      ; ### attribute# ## #### ### ### ###.
                                                    ;# #### #### #### Multi-Value# ##.
                                                    ;# ##### ## attribute# ### ##.
         [ "COLLECTIVE" whsp ]          ; default not collective
         [ "NO-USER-MODIFICATION" whsp ]; ### # attribute## #### ### ####. # #### ## ####
###### ##.
         [ "USAGE" whsp AttributeUsage ]; default userApplications
         whsp ")"

      AttributeUsage =
          "userApplications"     /
          "directoryOperation"   /
          "distributedOperation" / ; DSA-shared
          "dSAOperation"          ; DSA-specific, value depends on server
```

상당히 복잡해 보여서 질릴지도 모르겠지만 BNF를 보면서 하나 하나 정의를 해나간다면 그리 복잡하지는 않게 느낄수가 있다. 위에서 whsp란 공백(' ')문자를 의미하고 numericoid는 숫자로 구성된 ObjectIDentifier(OID)를 말한다. OID에 대해서는 따로 설명을 해두었으니 아직 읽지 않았다면 지금 읽어보기 바란다. qdescrs란 하나 또는 하나 이상의 이름이 위치한다는 뜻이다. qdescrs의 예를 들면 'cn'처럼 하나를 쓸수도 있고 두 개 이상의 이름을 위치시킬 필요가 있을때는 ('cn' $ 'commonName')처럼 '$'문자로 구분을 두고 쓸 수가 있다. 그리고 woid란 이름 또는 OID가 위치할수 있다는 뜻이다. 예를 들어 'cn'의 oid가 2.5.4.3이면 cn을 쓰거나 2.5.4.3을 쓸수가 있다는 뜻이다. 마지막으로 noidlen은 Length Specifier를 oid에 선택적으로 포함시킬수 있다는 것(예를 들어 1.3.6.1.4.1.1466.115.121.1.1510)으로 데이터의 타입과 길이를 정한다.
다음 두 개의 표는 각각 Equality, Ordering, Substr에 사용되는 Matching Rule과 Syntax에 사용될수 있는 항목들을 나타낸다.

OpenLDAP에서 지원하는 Syntax

Name	OID	Description
binary	1.3.6.1.4.1.1466.115.121.1.5	BER/DER data
boolean	1.3.6.1.4.1.1466.115.121.1.7	boolean value
distinguishedName	1.3.6.1.4.1.1466.115.121.1.12	DN
directoryString	1.3.6.1.4.1.1466.115.121.1.15	UTF-8 string
IA5String	1.3.6.1.4.1.1466.115.121.1.26	ASCII string
Integer	1.3.6.1.4.1.1466.115.121.1.27	integer
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	DN plus UID
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	numeric string
OID	1.3.6.1.4.1.1466.115.121.1.38	object identifier
Octet String	1.3.6.1.4.1.1466.115.121.1.40	arbitary octets
Printable String	1.3.6.1.4.1.1466.115.121.1.44	printable string

OpenLDAP에서 지원하는 Matching Rule

Name	Type	Description
booleanMatch	equality	boolean
objectIdentiferMatch	equality	OID
distinguishedNameMatch	equality	DN
uniqueMemberMatch	equality	DN with optional UID
numericStringMatch	equality	numerical
numericStringOrderingMatch	ordering	numerical
numericStringSubstringsMatch	substrings	numerical
caseIgnoreMatch	equality	case insensitive, space insensitive
caseIgnoreOrderingMatch	ordering	case insensitive, space insensitive
caseIgnoreSubstringsMatch	substrings	case insensitive, space insensitive
caseExactMatch	equality	case sensitive, space insensitive
caseExactOrderingMatch	ordering	case sensitive, space insensitive
caseExactSubstringsMatch	substrings	case sensitive, space insensitive
caseIgnoreIA5Match	equality	case insensitive, space insensitive
caseIgnoreIA5OrderingMatch	ordering	case insensitive, space insensitive
caseIgnoreIA5SubstringsMatch	substrings	case insensitive, space insensitive
caseExactIA5Match	equality	case sensitive, space insensitive
caseExactIA5OrderingMatch	ordering	case sensitive, space insensitive
caseExactIA5SubstringsMatch	substrings	case sensitive, space insensitive

이제 웬만큼 attribute를 정의하는데는 이해가 갔으리라 생각한다. 다음 예제를 한번 보자. 다음의 예제는 core.schema파일에 있는 attribute의 정의이다.

```
attributeType ( 2.5.4.41 NAME 'name'
                EQUALITY caseIgnoreMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.1532768 )
        attributeType ( 2.5.4.3 NAME ( 'cn' $ 'commonName' )
                SUP name )
```

두 개의 attribute가 정의되었음을 알수 있다. 먼저 정의된 name이라는 이름의 attribute는 equality와 substr에 대한 매칭 조건 속성을 가지면서 데이터 타입은 UTF-8 문자열 타입인 direcrtoryString형이고 최대 32768길이의 값을 가질수 있다. 다음에 정의된 attribute는 cn 또는 commonName이라 불리고 name attribute의 속성을 그대로 상속받는다.

ObjectClass 작성
잠깐 OID(Object Identifier)에 대해서 이야기를 해보자.

```
objectclass ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL
        MUST ( sn $ cn )
        MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )
```

앞에서 들었던 objectclass를 정의하는 예제이다. 여기에 2.5.6.6 이란 부분이 있는데 이것을 OID(Object IDentifier)라고 한다. LDAP v3에서는 이렇게 서로다른 LDAP 오브젝트들과 서로 구분될수 있도록 모든 요소에 OID를 부여한다. 마치 호스트에 고유한 아이피 번호를 부여하는 것과 마찬가지라고 생각할수 있다. 그리고 또한 도메인 네임을

부여했을 때 자신의 도메인 밑으로 계속 하부 도메인을 둘수 있는 것과 같이 자신에게 또는 그룹에게 할당받은 OID아래로 마음껏 확장하여 쓸수가 있다. 만일 자신에게 1.2.3.4 라는 OID가 부여되었다고 치자. 그러면 1.2.3.4 로 시작하는 모든 하부 OID를 자신이 마음대로 설정하여 쓸수가 있다. 1.2.3.4.1에는 attribute들을 할당하고 1.2.3.4.2를 objectclass들을 설정하고 등등...
그렇다면 이러한 OID를 어떻게 얻을 수가 있을까? IANA(Internet Assigned Numbers Authority)에서 private enterprise부문에 한해서 무료로 OID를 등록시켜주고 있다. 이러한 등록 서비스는 다음의 홈페이지에서 할 수가 있다.
http://www.iana.org/cgi-bin/enterprise.pl
현재 등록된 OID의 리스트는 다음의 홈페이지에서 볼수가 있다.
ftp://ftp.isi.edu/in-notes/iana/assignments/enterprise-numbers
대외적으로 쓰이는(혼자 쓰려는 것이 아닌) 엔터프라이즈급 LDAP서버라면 여기에 등록시키고 쓰길 바란다.

보안

네트워크 보안

- slapd 는 389 포트(ldap://), 636포트(ldaps://)를 사용한다. 현재 사용중인 방화벽에서 필요한 포트를 추가설정한다.
- slpad 는 tcp 래퍼를 지원한다.

데이터 무결성, 기밀성 보호

- 보안을 위하여 Transport Layer Security(TLS)를 사용한다. openldap 문서에서 TLS 부분을 참고한다.

인증 방법

- simple method : anonymous(no name, no password) / unauthenticated, and (name, no password) / user/password authenticated
- SASL : openldap 문서에서 SASL 참고

분산 디렉토리 서비스 구현

Subordinate Knowledge Information

- 하위트리에 대한 검색을 다른 디렉토리 서버로 위임

Superior Knowledge Information

- 상위 디렉토리 서버에 대한 검색

리플리케이션

- 1 master / n slave 구조

OpenLDAP의 관리

BackUp(백업)

- 첫째로 가장 간단하고 무식한 방법은 data디렉토리를 tar로 묶어서 보존하는 것이다. 그런다음 ldap 데몬들을 내리고 백업한 tar파일을 다시 풀어서 덮어쓴다음 다시 ldap데몬을 시작 시키면 이전의 내용이 그대로 들어 있다.
- 두번째로 오프라인 상태에서 slapcat 명령어를 이용하는 것이다. 참고로 rpm 설치시 데이터 디렉토리는 /var/lib/ldap 이며 아무런 데이터가 없는 상태에서 slapadd를 root로 실행할 경우 /var/lib/ldap 하위 디렉토리에 만들어진 파일은 root 권한을 가지고 있으므로 소유주를 ldap 으로 바꾸어주어야 한다.

백업받기

```
# slapcat -b "dc=samjung,dc=com" -l contents.ldif
```

백업자료를 이용하여 복원하기

```
# slapadd -l contents.ldif
```

퍼포먼스를 늘리기위한 방법들.

a. 인덱싱을 잘해야 한다. 인덱싱이 무조건적으로 많다고 해서도 안되며 필요한데도 쓰지 않는것도 안된다. 꼭 필요한 attribute에 한한 인덱싱을 해야 하며 그 attribute의 쓰임새에 맞는 인덱싱만을 해야 할것이다.
b. log를 줄임으로써 속도 향상을 꾀한다. slapd.conf에 'loglevel 0'을 명시하므로써 syslogd의 cpu사용을 줄일 수 있다. 한예로 다음과 같은 syslog.conf를 설정하게되면

```
# LDAP logs
LOCAL4.*                                -/var/log/ldap
```

펜티엄II 450Mhz머신에서 sn검색을 하게될 경우 3/sec걸리던 것이 51/sec로 늘어난다고 한다.

c. cachesize와 dbcachesize설정을 잘 해야한다.

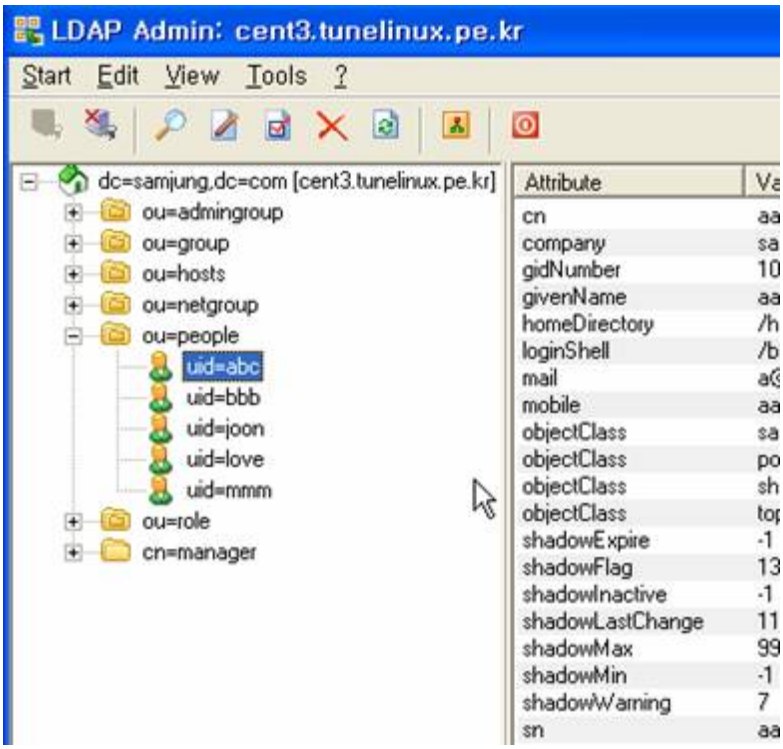
d. 퍼포먼스에 영향을 주는 시스템 요소로 메모리,디스크,파일시스템이 있다. 메모리를 늘린다든지 디스크를 빠른 고성능으로 교체한다든지 파일에 대한 timestamp수정,접근을 막는 설정을 하므로써 좀더 퍼포먼스를 향상 시킬수 있다.

그외의 자세한 부분은 http://www.openldap.org/faq/index.cgi?_highlightWords=performance&file=190 을 참조하도록 한다.

LDAP GUI 관리툴로 쉽게 관리할 수는 없을까?

LDAP의 관리를 보기에도 편하고 이용하기에도 편한 GUI기반 응용프로그램을 사용하여 한다면 좋을 것이다. 여기에 그 몇가지 유용한 GUI툴 리스트가 있으니 관리 환경을 바꿔보는것도 좋겠다.

-



ldap 검색, 수정 등 할 수 있는 윈도우 공개프로그램(GPL) 이 실제 사용해보니 편리함. GUI에서 사용자 이동, 복사, 그룹에 여러 사용자 추가등 가능함.
단점은 이 프로그램에서 한글을 입력할 경우 euc_kr 로 들어가기 때문에 UTF8로 볼 경우에는 한글이 깨져 보인다.
관리툴을 별도 제작하지 않는 경우 phpLDAPadmin을 쓰면 될 듯 하다.

- phpLDAPadmin
  http://ldapadmin.sourceforge.net 소스포지에 가장 최신 버전이 올라가있음
  LDAP 입력, 수정, 삭제
- LDAP Account Manager: lam.sourceforge.net
- LDAP Browser: www-unix.mcs.anl.gov/~gawor/ldap

FAQ(Frequently Asked Questions)

기존 문서에서 필요하다고 생각되는 것만 다시 뽑았다. 다음의 Q & A는 모두 데이터베이스 사랑넷의 LDAP게시판에서 오갔던 질문,답변중에서 몇가지를 골라 옮긴것이다. 질문과 답변을 주신 모든 분들(특히 최영봉님,김경하님,송상준님,정재익님,이기동님,임오근님,백종규님,양미향님)에게 감사드리고 싶고 별다른 허락없이 글을 여기에 옮기는 것에 대해 양해를 구하고 싶다.

Q. LDAP을 사용하면 구체적으로 좋은 점이 무엇인가요?
A. LDAP 과 DBMS 의 차이점을 설명드리면 답이 되겠군요. 한마디로 LDAP은 '특화된 데이터베이스'라고 볼 수 있습니다. 우선, LDAP은 쓰기 보다는 '읽기'에 최적화된 데이터베이스입니다. 따라서, 전화번호부처럼 색인을 이용하여 빠르게 원하는 정보는 찾을 필요가 있는 자료에 적합합니다. 예약시스템이나 게시판과 같이 수시로 변경되는 자료는 DBMS를 사용하는 것이 좋겠죠. 또, 한 번에 한 개의 정보를 찾는데 적합합니다. 이 때문에 LDAP에는 트랜잭션과 같은 개념이 없습니다. 은행 업무 처리 등에 사용하기에는 적합하지 않다는 뜻이죠. 주로, 인력 정보 시스템, DNS, 검색엔진의 인덱스, 주소록, SMTP가 사용하는 메일 룩업 테이블(Sendmail, Qmail, Postfix), PAM 등에 응용되고 있습니다. 아참, M$의 ActiveDirectory 도 들 수 있겠군요. NT5.0(윈2000)에서는 레지스트리, 계정관리를 이걸로 할 수 있도록 되어 있더군요.(약간 변종이긴 하지만 LDAP에 뿌리를 두고 있죠)
DBMS에서는 SQL 이라는 표준화되고 강력한 질의 언어를 제공함으로써 복잡하고 방대한 자료 가공 작업을 수행하기에 적합하지만, LDAP은 단순하면서도 속도에 최적화된 프로토콜을 사용함으로써 간단한 몇 개의 조건 비교를 통해 원하는 몇 가지 결과를 빠르게 얻는데 적합합니다.
결국, LDAP을 쓴다고 해서 무조건 좋아진다든가 하는 점은 없습니다. LDAP을 어디에 어떻게 사용하는지가 중요하죠. DBMS도 마찬가지입니다. 엉뚱한 곳에 사용하면 배보다 배꼽이 커지기도 하죠.

Q. LDAP의 모든 데이터를 삭제하려면 어떻게 하면 됩니까?
A. 간단히 데이터 디렉토리의 모든 파일을 지우면 모든 데이터가 삭제된 것입니다.

Q. 문서를 보다보면 N+1 Directory problem이란 말이 나오는데 이것이 무엇이고 무엇 때문에 발생하는지 알고싶습니다.
A. n+1 directory problem 이란 LDAP의 필요성에 대해 이야기할때 주로 나오는 말로 매니지먼트에 관련한 해결책이 LDAP로 이루어진다라는 결론으로 나갑니다. 우선 어떤 문제인지 예를 하나 들어 보겠습니다. 우리가 어떤 응용프로그램을 설치하게 되면 우리는 그 응용프로그램에 포함된 특정 설정파일을 설정해야 하는 일을 떠안게 됩니다. 센드메일을 설치하게되면 설정파일을 건드려줘야 하는것과 같습니다. 자 이제 기업환경으로 예를 좀더 구체화 시켜보겠습니다. 신입사원이 한명 회사에 입사를 하였습니다. 그러면 이 사람에 관련한 모든 설정을 관리자가 해줍니다.(메일계정설정, 데이타 디렉토리 접근권한 설정, 게시판 접근권한 설정 등등등 많겠지요?) 이제 이 유저(신입사원)의 정보를 변경시켜야 하는 경우가 생긴 다고 치면 참 관리자 입장에서는 할일이 아닐거라고 생각합니다. 규모가 큰 기업일 수록 더욱 관리가 힘들겠지요. 위의 경우 응용프로그램이 새로 하나 더 깔리게되면 시스템 관리자는 그만큼 하나더 늘어난 설정파일을 관리해야 합니다. 이것을 두고 N+1 Directory Problem 이라고 합니다. 이것을 해결하기 위해 위의 예제에서 모든 응용프로그램은 사용자의 설정 정보를 각각의 응용프로그램에서 읽어오지 않고 LDAP디렉토리 서버에서 읽어온다고 생각해 봅시다. 이제 관리자의 노가다는 줄어들겁니다. 한편 개발자의 입장에서도 아주 편해졌습니다. 이전까지는 해당 어플리케이션에 관련한 프로그램을 개발하기 위해 특정 프로그램의 스펙을 보고 따라해야 했습니다. 하지만 이제는 디렉토리서버 에서 유저와 그룹정보를 읽어오는 부분이 공통이 되었습니다. 그만큼 재사용성 면에서도 나아졌습니다. (딴말이지만 자바쪽에서는 LDAP를 EJB객체를 검색하기위한 도구로 사용하고 있고 인증도 LDAP를 사용하고 있습니다. 서로 다른 환경의 EJB서버에 대한 특정의 프로그램을 짜느니 LDAP-JNDI를 통한 일관된 접속으로 네트웍에 분산된 객체들을 빠르고 효율적으로 이용합니다.) 또다른 예제를 들어보면 위의 설정파일들은 대개 한곳에 모아놓습니다.(유닉스의 /etc디렉토리처럼) 그러나 한개의 서버라면 관리자에게 별일아닐수 있으나 개념을 확장해서 수십대의 서버를 가진 기업이라면 어떨까요? (이 가상의 예제는 책에서 봤지만) 수십대의 서버에 모두 아파치 웹서버가 설치되어 있다고 생각을 해봅시다. 시스템관리자가 이것을 모두 어떻게 설정을 할까요? 하나하나 하는 수 밖에 없을겁니다. 물론 비상한 관리자는 rdist,rsync같은 프로그램을 쓸수도 있습니다. 하지만 이것자체도 시스템관리자가 관리해야 할 것으로 늘어나죠. 하지만 이 설정파일을 LDAP에 넣고 일괄 관리 할수 있게 아파치가 프로그램된다면(설정파일을 LDAP에서 읽어오도록) 관리적인 측면에서 많은 이익을 얻을수 있을것이라 생각됩니다.

Q. ldap에 대해 한 3일 공부한 초보입니다. 막상 돌려보려구 하니까, 설정을 어떻게 해야될지... 깜깜하군요. suffix, rootdn 이 부분을 잘 몰르겠습니다. 구체적으로 어떻게 설정을 하면되겠습니까?
A. suffix는 DIT(Directory Information Tree) 구조상 최상위 Entry를 말합니다. 즉, c=KR이라는 Entry를 최상위 Entry로 해서 그밑에 Entry들을 달고 싶다고 하신다면, suffix는 c=KR이 되겠죠. 그외에도, o=My Company라든지, o=My Company,c=KR 과 같은 형태도 가능합니다. rootdn은 UNIX의 Root계정과 같이 LDAP서버에 대해 모든 권한을 가지는 사용자를 말합니다. UNIX나 DB와는 달리 모든 사용자는 DN형태로 존재해야 합니다. slapd.conf에는 원하는 DN을 적어주면 됩니다. 보통, cn=Directory Manager를 많이 씁니다.

Q. suffix와 rootdn의 차이점을 잘 모르겠습니다. DIT상에서 suffix는 최상의 entry인것 같은데, 그러면 rootdn은 실제적으로 무엇을 의미하는 지요?
A. rootdn 이란 말그대로 LDAP서버의 관리자의 distinguished name이라는 말입니다. LDAP에서는 이렇게 관리자도 dn형식으로 저장을 합니다. 예전에 잠깐 봤었는데 LDAP서버의 디렉토리가 방대하고 커서 혼자 관리하기 힘들다고 본다면 최상위 관리자 아래에 부서별로 관리자를 두고 엑세스 권한을 설정하고 하는 부분이 있더군요. openldap에서 검색을 해보시면 아마도 원하시는 부분을 좀더 자세히 이해하시리라 믿습니다.

PHP 샘플예제

관련자료

- PHP 매뉴얼에서 LDAP 부분 참고

LDAP 프로그래밍 작동 방식

- LDAP 프로그래밍을 하기 전에 알아야 할 부분
  디렉토리 서버의 이름이나 IP 주소

디렉토리 서버의 base dn (예를 들어 "dc=sample,dc=com")
서버에 접근하기 위한 dn과 비밀번호 (익명 사용자를 허용하지 않으면 접속할 dn과 비밀번호가 있어야 한다)

- 애플리케이션에서 LDAP 콜은 다음과 같은 형태로 이루어진다.

```
ldap_connect()    // ## ##
    |
 ldap_bind()       // #### ## ### ###
    |
#### ##, ####, ## #### #
 and display the results
    |
 ldap_close()      // ####
```

검색하기

```
<?php
/*********************************
LDAP search
*********************************/
// ### ####
$ldaphost="ldap://intranet.sds.co.kr/";
// ### ### dn ##
$userdn="cn=manager,dc=sds,dc=co,dc=kr";
// ####
$password="xxxxxx";

// ####
$ds=ldap_connect($ldaphost) or die("Could not connect to {$ldaphost}");  // must be a valid
LDAP server!

// id, password #### bind
$r=ldap_bind($ds,$userdn,$password) or die("Bind error");

// ### attr ##. ### ## ## ## attr# ## ###. ### ### ### attr# ###### ##
$justthese = array("uid","dn","cn", "mail",
"gidNumber","uidNumber","o","pager","shadowExpire","userpassword");

// ### ### ##
// ldap# ##### ## ##### ###
//$person="*joo*";
//$filter="(|(sn=$person)(uid=$person))"; // sn ## uid # $person ### #### ##.

//$filter="(&(|(sn=$person)(uid=$person))(objectclass=inetOrgPerson))";
//  sn ## uid # $person ### #### (or), objectclass# inetorgperson # ##. | # or, & # and ###.
$filter="objectClass=posixAccount";

// ####
$sr=ldap_search($ds, "ou=people,dc=sds,dc=co,dc=kr", $filter, $justthese);

// ## ###
echo "Number of entires returned is " . ldap_count_entries($ds, $sr) . "<br />n";

// #### ####
$info = ldap_get_entries($ds, $sr);
echo "Data for " . $info["count"] . " items returned:<p>nnn";

// #### ##
// ### ### ## attr ### ##### ##### #### ## ##### #### #####.
for ($i=0; $i<$info["count"]; $i++) {
    echo "uid : " . $info[$i]["uid"][0] . "<br />n";
    echo "dn : " . $info[$i]["dn"] . "<br />n";
    echo "cn : " . $info[$i]["cn"][0] . "<br />n";
    echo "email : " . $info[$i]["mail"][0] . "<br /><hr />n";
    echo "gidNumber : " . $info[$i]["gidnumber"][0] . "<br /><hr />n";
    echo "uidNumber ".$info[$i]["uidnumber"][0]."n";
    echo "userPassword ".$info[$i]["userpassword"][0]."n";
    echo "nn";
}

// ldap ####
ldap_close($ds);
```

```
?>
```

입력/수정하기

입력하는 경우 ldap_add 함수를 사용하고 수정하는 경우에는 ldap_modify를 이용한다. ldap_modify의 경우 기존에 정보가 있어야 한다.

```
<?php
$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {
    // bind with appropriate dn to give update access
    $r=ldap_bind($ds, "cn=manager, dc=samjung,dc=com", "xxxxx");

    // prepare data
    $info["cn"]="jjoon";
    $info["sn"]="jjoon";
    $info["objectclass"][0]="person";
    $info["objectclass"][1]="top";
    $info["telephoneNumber"]="02-2222";

    // add data to directory
    $r=ldap_add($ds, "cn=jjoon,ou=people,dc=samjung,dc=com", $info);
    //$r=ldap_modify($ds, "cn=jjoon,ou=people,dc=samjung,dc=com", $info);

    ldap_close($ds);
} else {
    echo "Unable to connect to LDAP server";
}
?>
```

삭제하기

```
<?php
$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {
    // bind with appropriate dn to give update access
    $r=ldap_bind($ds, "cn=manager, dc=samjung,dc=com", "xxxxxx");

    // delete data from directory
    $r=ldap_delete($ds, "cn=mytest3,ou=people,dc=samjung,dc=com");

    ldap_close($ds);
} else {
    echo "Unable to connect to LDAP server";
}
?>
```

SSL/TLS 사용하여 php프로그래밍하기

- SSL/TLS에 대하여
  SSL과 TLS는 다른 부분입니다.
  오고가는 패킷을 암호화하는 것은 SSL/TLS 두가지 방법이 있습니다.

오렐리 LDAP 서적 26쪽에 나와있는 내용입니다.

- LDAP over SSL (LDAPS – 636 port) 일반적으로 말하는 SSL입니다.

- TLS : 389 PORT 로 암호화해서 통신하는 기능으로 RFC 2830 에서 추가된 기능이라고 합니다. 389 동일한 포트에서 암호화 되지 않은 패킷과 암호화된 패킷을 동시에 전송할 수 있는 기능입니다.

- SSL/TLS 사용시 php로 프로그래밍을 할 경우 주의사항
  php에는 ssl 을 지원하는 636 port 로 접속할 경우 host 에 ldaps 로 지정하고 포트를 636 지정합니다. start_tls는 명시하지 않습니다. 오히려 명시하면 에러가 나지요.

TLS를 이용하는 경우는 389 PORT 자체적으로 암호화를 지원하기 때문에 389 포트를 이용합니다. PHP 의 경우는 기본적으로 프로토콜 2를 사용하기 때문에 프로토콜 3을 사용하도록 옵션을 세팅하고 호스트에서는 ldaps 가 아닌 ldap으로 지정을 합니다. 반드시 start_tls 옵션을 사용해야지요.

ldaps 와 start_tls 기능은 별개입니다.

그리고 SSL이나 TLS를 이용하는 경우 접속하려는 호스트에서 서버의 인증서가 있어야 합니다. 이는 아파치에서 ldap 인증에 SSL을 이용하는 경우와 동일하게 설정을 해야 한다. 아래에서 CA키는 실제 사용하는 것으로 바꾸면 된다.

```
LDAPTrustedCA /etc/openldap/test/mirror.crt
LDAPTrustedCAType BASE64_FILE
```

다시 정리하면 다음과 같습니다.

- PHP에서는 기본적으로 프로토콜 2를 사용하기 때문에 프로토콜 3을 사용하도록 옵션을 세팅
- SSL을 사용할 경우 636 포트를 사용하고 서버는 ldaps:// 으로 지정합니다.
- TLS를 사용할 경우 389 포트를 사용하고 서버는 ldap:// 으로 지정합니다. ldap_start_tls 를 지정해줘야 사용이 가능합니다.
- ldap_start_tls 는 ldap 서버에 연결(connect)한후 바인드(ldap_bind) 하기전에 지정합니다.
- 포트를 지정하지 않을 경우 SSL이냐 TLS냐에 따라 자동으로 포트를 선택합니다.

참고자료 : php 에서 start_tls 사용하기 http://kr.php.net/ldap_start_tls

아래 코드에서 차이가 나는 부분은 프로토콜을 3으로 세팅하고 ldap_start_tls를 해주는 부분이다.

```
if ($ds) {
   if (!ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3)) {
     fatal_error("Failed to set LDAP Protocol version to 3, TLS not supported.");
   }
}

if (!ldap_start_tls($ds)) {
    echo "failed start_tls\n";
    exit;
}
```

위의 내용을 반영한 전체코드이다.

```
<?php
/**********************************
LDAP search
**********************************/
// ### ####
$ldaphost="ldap://intranet.sds.co.kr/";
// ### ### dn ##
$userdn="cn=aareadonly,ou=admingroup,dc=sds,dc=co,dc=kr";
// ####
$password="xxxxx";

// ####
$ds=ldap_connect($ldaphost) or die("Could not connect to {$ldaphost}");  // must be a valid
```

```
LDAP server!

if ($ds) {
   if (!ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3)) {
     fatal_error("Failed to set LDAP Protocol version to 3, TLS not supported.");
   }
}

if (!ldap_start_tls($ds)) {
    echo "failed start_tls\n";
    exit;
}

// id, password #### bind
$r=ldap_bind($ds,$userdn,$password) or die("Bind error");

// ### attr ##. ### ## ## ## attr# ## ###. ### ### ### attr# ###### ##
$justthese = array("uid","dn","cn", "mail",
"gidNumber","uidNumber","o","pager","shadowExpire","userpassword");

// ### ### ##
// ldap# ##### ## ##### ###
//$person="*joo*";
//$filter="(|(sn=$person)(uid=$person))"; // sn ## uid # $person ### #### ##.

//$filter="(&(|(sn=$person)(uid=$person))(objectclass=inetOrgPerson))";
//  sn ## uid # $person ### #### (or), objectclass# inetorgperson # ##. | # or, & # and ###.
$filter="objectClass=posixAccount";

// ####
$sr=ldap_search($ds, "ou=people,dc=sds,dc=co,dc=kr", $filter, $justthese);

// ## ###
echo "Number of entires returned is " . ldap_count_entries($ds, $sr) . "<br />n";

// #### ####
$info = ldap_get_entries($ds, $sr);
echo "Data for " . $info["count"] . " items returned:<p>nnn";


// #### ##
// ### ### ## attr ### ##### ##### #### ## ##### #### #####.
for ($i=0; $i<$info["count"]; $i++) {
    echo "uid : " . $info[$i]["uid"][0] . "<br />n";
    echo "dn : " . $info[$i]["dn"] . "<br />n";
    echo "cn : " . $info[$i]["cn"][0] . "<br />n";
    echo "email : " . $info[$i]["mail"][0] . "<br /><hr />n";
    echo "gidNumber : " . $info[$i]["gidnumber"][0] . "<br /><hr />n";
    echo "uidNumber ".$info[$i]["uidnumber"][0]."n";
    echo "userPassword ".$info[$i]["userpassword"][0]."n";
    echo "nn";
}

// ldap ####
ldap_close($ds);

?>
```