

TD4

LDAP Lightweight Directory Access Protocol

Manipulations dans le cadre du cours de Gestion et Administration des Réseaux.

Mise en œuvre des concepts d'annuaire LDAP via **OpenLDAP** (<http://openldap.org>).

OpenLDAP est une implémentation libre d'un annuaire LDAP. Dans la suite vous aurez l'opportunité d'installer et de configurer un serveur LDAP. Nous n'aborderons pas dans ce td les aspects **sécuritaire** de ldap¹. Il faudrait parler de "tunnel SSH", de cryptographie via les algorithmes MD5, SSHA, ...

Je profite de ce résumé pour remercier les auteurs de [1] dont les éclaircissements m'ont beaucoup apporté. Les étudiants de la promotion 2004-2005 ont aussi apporté leur pierre à l'édifice en donnant les exemples de manipulation des diverses API. Qu'ils en soient ici remerciés. Selon la formule consacrée maintenant au sein de l'école j'en profite pour remercier ici, tous ceux qui aiment être remerciés.

Organisation du travail

Vous travaillerez de manière individuelle pour ces manipulations. N'oubliez pas de consulter la liste de lecture avant et pendant ces manipulations.

Vous remettrez un document reprenant vos diverses manipulations au terme de ce travail ... comme vous en avez l'habitude maintenant.

Installation de OpenLDAP

Installation des paquets de la distribution. Sous *debian*, installez les paquets `slapd`, `ldap-utils` et `openssl`².

¹Le *planning* ne nous en laissant pas trop le temps

²Ce paquet sera utilisé pour crypter les transmissions sur le réseau

Lors de la configuration du `daemon slapd`, choisissez la configuration **auto** avec les paramètres `custom` pour le choix du suffix. Vous choisirez `dc=esi,dc=be`. Ceci aura pour effet de générer un fichier de configuration de `slapd` minimal.

Dans la suite, il faut distinguer le fichier `/etc/ldap/slapd.conf` configurant le serveur et utile aux outils associés tels que `slapdcat`, `slapdadd` et le fichier `/etc/ldap/ldap.conf` utile aux clients ldap tels que `ldapadd`, ...

1 Configuration slapd.conf

1.1 À propos des schémas

Pour configurer son serveur ldap, il faut mettre en place des **schémas**. Certains existent et sont fournis avec l'implémentation d'autres peuvent être écrit. Dans une première approche, nous utiliserons les schémas existants. Notre manipulation portera sur le schéma `inetOrgPerson`.

L'inclusion des divers schémas se fera par

```
# Schema and objectClass definitions
include      /etc/ldap/schema/core.schema
include      /etc/ldap/schema/cosine.schema
include      /etc/ldap/schema/inetorgperson.schema
```

1.2 À propos des log

Il est toujours bon d'avoir des *logs* (surtout en phase de test) de fonctionnement d'un serveur. Nous pouvons préciser au serveur la "quantité" d'informations que nous désirons. Pour ce faire on positionne le paramètre `loglevel`. Les différentes valeurs peuvent être vue comme une série de choix que nous pouvons additionner. Nous choisissons 296, soit 8+32+256, gestion des connexions, traitement des filtres de recherche et statistiques sur les connexions ... voir tableau 1.

Nous ajoutons à notre fichier de configuration les lignes où `pidfile` caractérise le fichier contenant le PID du *process* LDAP et `argsfile` caractérise le fichier contenant les options de serveur.

```
# Log
loglevel     296
pidfile      /var/run/slapd.pid
argsfile     /var/run/slapd.args
```

Niveau	Informations enregistrées
-1	Toutes les informations
0	Aucune information
1	Trace les appels de fonctions
2	Informations de débogages liées au traitement des paquets
4	Débogage verbeux
8	Gestion des connexions
16	Paquets émis et reçus
32	Traitement des filtres de recherche
64	Traitement du fichier de configuration
128	Traitement des listes de contrôle d'accès
256	Statistiques sur les connexions, les opérations et les résultats
512	Statistiques sur les résultats renvoyés aux clients
1024	Communications avec les BDs de type shell
2048	Affichage des informations de débogage sur les traitements des entrées

TAB. 1 – Niveaux de journalisation d'OpenLDAP

1.3 À propos du cryptage des *password*

Les mots de passe ne seront probablement pas stockés en clair dans l'annuaire. Ils peuvent être cryptés au moyen de divers algorithmes ; SSHA, MD5, ... Par défaut, nous utiliserons le cryptage SSHA.

Nous ajoutons au fichier de configuration

```
#Security
password-hash {SSHA}
```

1.4 À propos des bases de données

Après la section globale se trouve(nt) une ou plusieurs sections consacrées aux bases de données. Nous n'en définirons qu'une. Le premier paramètre définit le type de BD à utiliser. Nous utiliserons **bdb** (car il est proposé par notre distribution).

Il est également utile de définir le contexte de nommage ou *suffix* et l'endroit où sera stockée la BD. A ce stade, nous ajoutons dans le fichier de configuration,

```
#####
# database definitions
#####
# Database type
database bdb
```

```
# The base of your directory
suffix "dc=esi,dc=be"
# Where the database file are physically stored
directory "/var/lib/ldap"
mode 0600
```

Un annuaire LDAP peut avoir un utilisateur *root*, un administrateur de l'annuaire ayant tout les droits. Les restrictions d'accès à la base de données n'ayant pas d'effet sur cet utilisateur. Certains administrateurs préfèrent configurer ce **DN root** uniquement lors de la mise au point de l'annuaire. Un tel utilisateur est défini grâce à

```
# DN root definition
rootdn "cn=admin,_dc=esi,_dc=be"
rootpw {SSHA}8yBWFWR770qXIBf8G1fs/GLp23G5axHa
```

Le mot de passe crypté est défini grâce à l'utilitaire `slappasswd`. Une commande du style suivant devrait le générer

```
slappasswd -s secret
```

1.4.1 Indexation de la BD

Généralement, on indexe une BD afin de pouvoir accéder rapidement à l'information. Le paramètre *index* définit pour quels attributs `slapd` doit gérer un *index*. `Slapd` connaît quatre types d'index.

- ↪ *approx (approximate)* - optimise les recherches approximatives, voire phonétiques
- ↪ *eq (equality)* - optimise les recherches exactes. La recherche pourra être sensible à la casse et/ou aux espaces suivant les règles de correspondances définies dans la syntaxe de l'attribut
- ↪ *pres (presence)* - permet de savoir si un attribut a une valeur
- ↪ *sub (substring)* - permet une recherche par sous-chaine

Pour des raisons de performances, *OpenLDAP* impose d'indexer 'objectClass'. Nous choisissons d'indexer l'attribut *cn* sur la "présence" et l'"égalité". Nous aurons

```
# Indexing options
index objectClass eq
index cn pres,eq
```

1.4.2 ACL

Les ACL *OpenLDAP* permettent de préciser *qui a accès à quoi* dans l'annuaire. Le *qui* peut être choisi parmi

- ↪ * - tout utilisateur connecté

- ↪ `self` - DN de l'utilisateur actuel
- ↪ `anonymous` - connexions non authentifiées
- ↪ `user` - connexions authentifiées
- ↪ `regexp` - l'expression régulière est comparée à un DN ou à un identifiant

Les niveaux d'accès sont, quant-à eux choisis parmi

- ↪ `write` - mise à jour des valeurs des attributs
- ↪ `read` - lecture des résultats de recherche
- ↪ `search` - application de filtre de recherche
- ↪ `auth` - authentification d'un utilisateur
- ↪ `none` - aucun accès

Le *quoi* détermine les entrées concernées par l'ACL. Il est composé de trois parties ; une expression régulière déterminant le DN, un filtre de recherche et une liste de noms d'attributs séparés par des virgules.

Les ACL suivantes définissent un accès en lecture par défaut. Elles permettent aux utilisateurs d'accéder à l'attribut `userPassword` à des fins d'authentification. Un utilisateur pourra modifier son propre mot de passe. Pour le reste, l'accès se fait en lecture.

```
#
# ACL
#
defaultaccess search

# Access userPasswd only for auth but user can modify own passwd
access to attrs=userPassword
      by self write
      by * read

# ACL anybody can read
access to *
      by * read
```

1.5 Démarrage du serveur

Afin de mettre en place un serveur LDAP, nous devrions analyser le type d'éléments que devrait contenir notre annuaire et choisir, voire écrire, nos schémas (voir la section 3.2). Nous utiliserons donc le schéma `inetOrgPerson`. Ce schéma est défini pour stocker des personnes. Il hérite de `person` (défini dans le fichier `core.schema`).

Avant de lancer notre serveur, il reste à créer le répertoire qui va contenir notre BD, dans notre cas `/var/lib/ldap`

```
# mkdir /var/lib/ldap
# chmod 700 /var/lib/ldap
```

À ce stade, nous pouvons démarrer le serveur LDAP à l'aide du script.

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

Vérifions via `ps` que le *daemon* tourne. Sur une machine Linux, ce sont, en réalité, trois daemons qui tournent.

```
$ ps aux | grep slapd
root 2325 0.0 2.8 7816 1760? Ss 14 :59 0 :00 /usr/sbin/slaped
root 2326 0.0 2.8 7816 1760? S 14 :59 0 :00 /usr/sbin/slaped
root 2327 0.0 2.8 7816 1760? S 14 :59 0 :00 /usr/sbin/slaped
```

Avant d'arrêter le serveur *OpenLDAP*, il faut laisser le daemon purger ses données. Pour ce faire, on envoie un signal d'interruption `INT` au serveur afin qu'il puisse s'arrêter convenablement. Il faut connaître le PID du process ... que l'on a judicieusement sauvegardé dans le fichier `/var/run/slaped.pid`. Nous exécutons la commande

```
# kill -INT `cat /var/run/slaped.pid`
```

Nous insistons donc, **ne pas arrêter brutalement un serveur ldap**, sinon vous risquez de compromettre vos données. Si un script existe (et c'est notre cas) utilisons-le pour arrêter le serveur

```
/etc/init.d/slaped stop
```

2 Fichiers LDIF et données de l'annuaire

Un annuaire sans données est un peu comme un ordinateur sans processeur, il ne sert pas à grand chose.

Il existe **deux manières** d'ajouter des données à un annuaire

- ↪ les utilitaires `slaped*`, permettent à l'administrateur d'importer les données directement dans la base de données. **Le serveur ne tourne pas**³
- ↪ les utilitaires `ldap*`, permettent de modifier l'annuaire "en ligne". **Le serveur tourne.**

2.1 Création de l'annuaire

Nous allons créer un annuaire contenant des *student*. Le fichier LDIF correspondant est le suivant.

```
# Creation du noeud racine de l'annuaire
dn:_dc=esi,dc=be
objectClass:_dcObject
```

³A partir d'OpenLDAP 2.1, on pourra relâcher cette contrainte

```
objectClass:_organizationalUnit
dc:_esi
ou:_Ecole_Superieure_dInformatique

#_Creation_de_l'OU_student
dn: ou=student,dc=esi,dc=be
objectClass: organizationalUnit
ou: student
```

Si le fichier s'appelle /tmp/racine.ldif, on l'inclura dans l'annuaire via la commande

```
# slapadd -v -l /tmp/racine.ldif
added : "dc=esi,dc=be" (00000001)
added : "ou=student,dc=esi,dc=be" (00000002)
```

Un slapcat permet de voir ce que contient la BD

```
# slapcat
dn : dc=esi,dc=be
objectClass : dcObject
objectClass : organizationalUnit
dc : esi
ou : Ecole Superieure dInformatique

dn : ou=student,dc=esi,dc=be
objectClass : organizationalUnit
ou : student
```

Nous pouvons lancer le serveur afin d'effectuer des tests "en ligne". Nous utiliserons les outils *ldap**.

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

Le serveur est opérationnel, nous pouvons utiliser les commandes ; ldapsearch, ldapmodify, ...

Commençons par consulter le contenu de tout l'annuaire⁴. Un ldapsearch comme celui-ci devrait suffire.

```
$ ldapsearch -x -b "dc=esi,dc=be" "(objectclass=*)"
-h 127.0.0.1
version : 2
#
# filter : (objectClass=*)
# requesting : ALL
#
```

⁴C'est-à-dire peu de choses

```
# esi, be
dn : dc=esi, dc=be
objectClass : dcObject
objectClass : organizationalUnit
dc : esi
ou : Ecole Superieure dInformatique

# student, esi, be
dn : ou=student, dc=esi, dc=be
objectClass : organizationalUnit
ou : student

# search result
search : 2
result : 0 Success

# numResponses : 3
# numEntries : 2
```

L'option *-x* indique que nous voulons une authentification simple⁵, l'option *-b* précise quel est le *suffix* de l'annuaire et *-h* renseigne quelle est la machine hôte.

Remarque Pour éviter de préciser l'adresse du serveur LDAP, vous pouvez ajouter la ligne suivante dans le fichier /etc/ldap/ldap.conf.

```
HOST 127.0.0.1
```

2.2 Mise à jour des données

Pour ajouter (ou modifier) des données, nous utilisons l'utilitaire *ldapmodify* avec "le fichier LDIF qui va bien". Voici un exemple de fichier LDIF permettant l'ajout d'utilisateurs.

```
# Ajout d'utilisateur

#_Entree_Juste_LEBLANC
dn:_cn=Juste_LEBLANC,ou=student,dc=esi,dc=be
objectclass:_inetOrgPerson
cn:_Juste_LEBLANC
sn:_Juste
mail:_juste.leblanc@dev.null
mail:_juste@leblanc.name
description:_il_s'appelle Juste Leblanc. Ah bon, il a pas de prenom.
```

⁵À l'inverse d'une authentification SASL dont nous n'avons pas parlé

```
# Entrée Marlene Sassoœur
dn: cn=Marlene_SASSOEUR,ou=student,dc=esi,dc=be
objectclass: inetOrgPerson
cn: Marlene_SASSOEUR
sn: Marlene
mail: marlene.sassoœur@dev.null
description: Elle me dit c'est_Marlene_sa_soeur._Avouez_que_c'est_confusant.
```

Si le fichier s'appelle /tmp/ajout.ldif, nous ajouterons les entrées dans l'annuaire via la commande

```
ldapmodify -D "cn=admin,dc=esi,dc=be" -W -x
-a -v -f /tmp/ajout.ldif
```

Les options signifient

- ↪ -D, précise qui est le *rootdn*
- ↪ -W, prompte pour un *password* (est semblable à -w secret)
- ↪ -x, indique l'authentification simple
- ↪ -a, précise qu'il s'agit d'un ajout
- ↪ -f, renseigne le fichier source, par défaut c'est *stdin*
- ↪ -v, verbeux

Pour la modification, il faudra préciser, via l'argument *changetype* le type de modification ; modification, suppression, ajout. Nous écrivons un fichier LDIF du genre.

```
# Ajout d'utilisateur

#_Entree_Juste_LEBLANC
dn: _cn=Juste_LEBLANC,ou=student,dc=esi,dc=be
changetype:_modify
add:_labeledURI
labeledURI:_http://juste.leblanc.name

#_Entree_Marlene_Sassoœur
dn: _cn=Marlene_SASSOEUR,ou=student,dc=esi,dc=be
changetype:_modify
delete:_mail
mail:_marlene.sassoœur@dev.null

#_Entree_Bidon
dn: _cn=Bidon,_ou=student,dc=esi,dc=be
changetype:_add
objectclass:_inetOrgPerson
cn:_Bidon
sn:_Bidon

#_Entree_Francois_Pignon
dn: _cn=Francois_PIGNON,ou=student,dc=esi,dc=be
changetype:_add
objectclass:_inetOrgPerson
cn:_Francois_PIGNON
sn:_Francois
description:_Grand_collectionneur_de_constructions_en_allumettes
```

```
#_Suppression_de_l'entree
dn: cn=Bidon,ou=student,dc=esi,dc=be
changetype: delete
```

Si le fichier s'appelle /tmp/modif.ldif, nous modifierons les entrées dans l'annuaire via la commande

```
ldapmodify -D "cn=admin,dc=esi,dc=be" -W -x
-v -f /tmp/modif.ldif
```

Remarque slapd suppose que les fichiers sont encodés en UTF8. Les données sont encodées en UTF8 et codées en Base64 afin d'éviter toute altération des données. Nos éditeurs ont tendance à utiliser les *charset* locaux et encoder les caractères en ISO8859-1 (voire ISO8859-15). Ceci ne pose pas de problème ... si l'on n'utilise pas nos chers accents. Dans cette hypothèse ceux-ci seront encodés par l'annuaire et retransmis de manière illisible ... un peu comme si la phrase contenant les caractères était "cryptée". Ces phrases sont reconnaissables aux :: suivant le nom de l'attribut.

L'utilitaire **iconv** permet de passer d'un encodage à l'autre. Il faudra donc convertir son fichier de données LDIF en utf8 avant de le "passer" aux utilitaires ldap*. Un peu comme ceci

```
iconv -f LATIN1 -t UTF8 monfichier_latin1.ldif >
monfichier_utf8.ldif
```

Lors de la restitution de données par ldap, celui-ci convertira les données en Base64 dès qu'il ne s'agit pas de caractère ASCII (il pense que ces données sont binaires). Il faudra donc rendre cette sortie lisible sur nos terminaux. Nous utiliserons le script `ldif2latin1`

```
ldapsearch -x -b "dc=esi,dc=be"
"(objectclass=*)" | ldif2latin1
```

Il faudra tenir compte de cet état de fait lors de la manipulation de l'annuaire via les APIs (voir 4).

2.3 Recherche dans l'annuaire

La recherche la plus globale est la recherche *montre-moi tout*. Une commande du genre devrait suffire (la seconde commande demande de lister uniquement les attributs dn et sn.

```
ldapsearch -x -b "dc=esi,dc=be" "(objectclass=*)"
ldapsearch -x -b "dc=esi,dc=be" "(objectclass=*)" dn
sn
```

Un filtre LDAP a la forme suivante

```
( attribut opérateur valeur )
```

L'attribut est le nom de l'attribut ! L'opérateur est choisi parmi

- ↪ = pour l'égalité,
- ↪ ~= pour les comparaisons approximatives,
- ↪ <= pour les comparaisons "inférieur ou égal",
- ↪ >= pour les comparaisons "supérieur ou égal"

La partie *valeur* peut être une valeur absolue (`cn=juste`) ou une valeur reposant sur les *wildcards* (`cn=*blanc`).

Nous pouvons regrouper des filtres élémentaires en utilisant les opérateurs booléens, & (ET), | (OU) et ! (NON). Ces opérateurs utilisent la notation *préfixée*. Pour rechercher les enregistrements ayant comme nom "gouyasse" ou "quitine", nous aurons

```
( | (cn=gouyasse) (cn=quitine) )
```

Pour rechercher des enregistrements des brasseries de Silly et d'Ellezelles qui sont des "pils", on aura

```
( & ( | (ou=*Ellezelle*) (ou=*Silly*) ) (cn=pils) )
```

Afin d'éviter d'obtenir trop de résultats lors d'une recherche du genre (`objectclass=*`), `ldapsearch` permet de définir des limites quant à l'information retournée.

- ↪ `-l entier`, définit en secondes, la durée d'attente maximale de la réponse à une demande de recherche.
- ↪ `-z entier`, définit le nombre d'entrées maximal à récupérer lorsqu'une recherche aboutit.

Ces valeurs peuvent-être définie dans le fichier `ldap.conf` en utilisant les paramètres ; `timelimit` et `sizelimit`. Une valeur de 0 en "ligne de commande" annule les limites imposées dans le fichier de configuration des utilitaires `ldap`.

3 Modification de schéma

Les schémas proposés par le serveur ne sont pas toujours adaptés. Pour créer un schéma personnel, il faut

- ↪ attribuer un OID unique aux nouveaux attributs et aux nouvelles classes.
- ↪ créer le fichier de schéma et l'"inclure" dans le fichier de configuration du serveur `slapd.conf`

Pour rappel, les schémas sont approuvés par l'IANA (voir [2]). Cet organisme attribue un **numéro d'entreprise unique** à qui en fait la demande (via

<http://www.iana.org/cgi-bin/enterprise.pl>). Le numéro attribué à l'école est 23162

Les OIDs doivent être uniques. Suivant [2] et [1], nous allouons l'arc d'OID 1.3.6.1.4.1.23162.504 aux nouvelles classes d'objets en suivant la structure ⁶

```
iso(1)
|- org(3)
|--- dod(6)
|---- internet(1)
|----- private(4)
|----- enterprise(1)
|----- esi.be(23162)
|----- local504(504)
```

Nous ne pouvons pas modifier la structure de l'arbre "au-dessus" de notre numéro d'entreprise mais nous sommes libres de l'organiser comme bon nous semble "en-dessous". Je me réserve le numéro 504 pour le local et laisse le reste libre pour les autres laboratoires. Nous utiliserons 23162.504.1 pour les classes d'objets et 23162.504.2 pour les attributs que nous créerons.

Nous décidons de créer un schéma permettant de stocker des informations sur des **bières** ⁷. Nous définirons un DN grâce au nom de la *brasserie* et à celui de la *bière*.

Pour définir la *brasserie* nous utiliserons le schéma existant `organizationalUnit` tandis que pour la *bière*, nous en créerons un pour l'occasion.

3.1 Schéma `organizationalUnit`

Ce schéma propose les attributs décrits dans le tableau 2. Il nous permettra de sauvegarder les données relatives à une brasserie. Dans un premier temps, un nom et une description suffiront.

3.2 Schéma `beerObject`

Nous aurons donc une classe d'objet `beerObject` ayant comme attributs obligatoires `cn` et `sn`, respectivement un *common name* et un *surname*. Les attributs facultatifs seront (par exemple)

- ↪ `alcoholVolume` - volume d'alcool en %
- ↪ `jpegPhoto` - photo de l'étiquette
- ↪ `conditionning` - conditionnement (bouteille de 25cl, 33cl, fût, ...)

⁶Je vous conseille vivement la lecture de [2] afin de parfaire votre compréhension de OID

⁷Ne croyez pas que je sois omnubilé par ça, c'est juste un choix d'exemples

↪ description

↪ ...

Le fichier schéma `beer.schema` aura l'allure suivante.

```
#
# Local definition of class beer
#
attributetype ( 1.3.6.1.4.1.23162.504.2.3 NAME 'alcoholVolume'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768}
)

attributetype ( 1.3.6.1.4.1.23162.504.2.5 NAME 'conditionnement'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768}
)

# beerObject
# beerObject represent one type of beer. There are a lot belgian beer.
# Each of them are represented with a name, degreeAlcohol ...
objectclass ( 1.3.6.1.4.1.23162.504.1.1 NAME 'beerObject'
    SUP top STRUCTURAL
    MUST ( cn $ sn )
    MAY ( alcoholVolume $ jpegPhoto $ conditionnement $ description )
)
```

Il reste à remplir l'annuaire de valeurs. Lorsque c'est fait, vous pouvez jouer avec l'annuaire ! Faites quelques recherches et vérifiez que vous avez judicieusement positionné vos *index* (voir le fichier de configuration `slapd.conf`).

Voici un fichier LDIF permettant de remplir l'annuaire^{8 9}.

```
# Creation du noeud racine de l'annuaire
dn:_dc=esi,dc=be
objectClass:_dcObject
objectClass:_organizationalUnit
dc:_esi
ou:_Ecole_Superieure_dInformatique

#_Creation_de_la_brasserie_Ellezelloise
dn:_ou="Brasserie_Ellezelloise",dc=esi,dc=be
objectClass:_organizationalUnit
ou:_Brasserie_Ellezelloise
postalAddress:_Rue_Guinaumont,_75,_7890_Ellezelles
seeAlso:_ou=brasseries,dc=namok,dc=be

#_Ajout_de_diverses_bieres...
#_Entree_"Blanche_des_saisais"
dn:_cn="Blanche_des_saisais",ou="Brasserie_Ellezelloise",dc=esi,dc=be
objectClass:_beerObject
cn:_Blanche_des_saisais
sn:_Blanche_des_saisais
description:_C'est une bière blonde refermentée en bouteille.
```

⁸Il contient très peu d'enregistrements, mais je compte sur vous pour en ajouter

⁹À ce stade — puisque vous connaissez la structure des fichiers LDIF — vous pouvez utiliser le *frontend gq* pour accéder à votre annuaire et y faire vos ajouts.

```
Elle se compose de 60% de malt d'orge_et_40%_de_froment
non_malté._Son_amertume;_que_du_houblon,_pas_de_coriandre,
pas_d'écorce_d'orange_à_mère."
alcoholVolume:_6.2%
```

```
#_...
```

Nous allons maintenant pouvoir interroger notre annuaire.

4 Accès via l'API *foo*

La plupart des langages proposent une API permettant d'interroger un serveur LDAP. Voici en **très** bref des moyens de le faire.

Vous testerez, au choix, l'API Perl ou PHP et cherchez à la faire dans un autre langage ...

4.1 Perl

Afin d'utiliser l'API perl, il est bon de disposer du paquet `Net::LDAP`. Vous trouverez ce package sur CPAN ou dans votre distribution (un `apt-get install libnet-ldap-perl` devrait suffire). Vous trouverez plus de détails dans [1], voici la base permettant une recherche du type *montre-moi tout*.

Vous testerez plus avant en ajoutant des données dans l'annuaire par exemple.

```
#!/usr/bin/perl

use Net::LDAP;
use Net::LDAP::LDIF;

my $HOST = "127.0.0.1" ;
## connexion au serveur et authentication
my $ldap = Net::LDAP->new($HOST, port => 389, version => 2) or die $!;
$ldap = Net::LDAP->new($HOST) or die $!;
$ldap->bind("cn=admin,ou=student,dc=esi,dc=be",password =>"system") or die $!;

## recherche "montre-moi tout"
my $msg = $ldap->search(
    base => "ou=student,dc=esi,dc=be",
    scope => "sub",
    filter => "(cn=*)",
    attrs => ["cn","mail"]
);

##affichage resultat
if ($msg->count()>0) {
    print $msg->count(),"entrées renvoyées.\n";
    foreach $entree ( $msg->all_entries() ) {
        $entree->dump();
    }
} else {
    printf ("pas_de_résultats\n");
}
```

```

}

##deconnect et sortie
$ldap->unbind();

```

4.2 PHP

L'API PHP est assez "semblable" à celle de perl. Elles sont toutes deux "orientée objet". Si PHP n'est pas installé, il faudra installer les paquets adéquats (un apt-get install php4 php4-cli php4-doc devrait suffire).

```

<?
$host="127.0.0.1" ;

// Connection
$sds=ldap_connect($host) or die("Connection_impossible_au_serveur_LDAP\n");
ldap_set_option($sds,LDAP_OPT_PROTOCOL_VERSION,3) ;

// Liaison avec un dn authentifié autorisant les modifications
$dn = "uid=reader,ou=users,dc=namok,dc=be";
$r=ldap_bind($sds, $dn, "secret") or die("Erreur_d'authentification\n");

// Ajout d'un champ "complet"
// $info["cn"]="Francois Pignon";
// $info["sn"]="Pignon";
// $info["mail"]="francois.pignon@finances.fgov.be";
// $info["objectclass"]="inetOrgPerson";
// $r=ldap_add($sds,
// "cn=Francois Pignon,ou=student,dc=esi,dc=be", $info)
// or die("Erreur d'ajout\n");

// Modification d'un enregistrement, ajout d'un attribut
// $info2["mail"]="marlene.sassoieur@dev.null.new";
// $r=ldap_modify($sds,
// "cn=Marlene Sassoieur,ou=student,dc=esi,dc=be", $info)
// or die("Erreur d'ajout\n");

// Recherche style 'montre-moi tout'
$filter = "sn=*";
$dn="dc=namok,dc=be";
$sr=ldap_search($sds, $dn, $filter);
echo "Le_résultat_de_la_recherche_est_:". $sr. "\n";
echo "Le_nombre_d'entrées_retournées_est_".ldap_count_entries($sds, $sr). "\n";
echo "Lecture_de_ces_entrées_...\n";
$info = ldap_get_entries($sds, $sr);
echo "Données_pour_". $info["count"]. "entrées:\n";

for ($i=0; $i < $info["count"]; $i++) {
    echo "ndn_est_:". $info[$i]["dn"]. "\n";
    echo "premiere_entree_cn_:". $info[$i]["cn"][0]. "\n";
    echo "premier_email_:". $info[$i]["mail"][0]. "\n";
    echo "object_class_:". $info[$i]["objectclass"][0]. "\n";
    echo "object_class_:". $info[$i]["objectclass"][1]. "\n";
}

echo "Fermeture_de_la_connexion\n";
ldap_close($sds);
?>

```

5 Pour aller plus loin

Il reste beaucoup d'aspects d' *OpenLDAP* que nous n'avons pas abordés. Il aurait fallu ¹⁰ traiter de la sécurité plus avant. Aborder le cryptage des données ainsi que l'accès à l'annuaire.

¹⁰L'an prochain sans doute

Table des matières

1	Configuration slapd.conf	2
1.1	À propos des schémas	2
1.2	À propos des <i>log</i>	2
1.3	À propos du cryptage des <i>password</i>	3
1.4	À propos des bases de données	3
1.4.1	Indexation de la BD	4
1.4.2	ACL	4
1.5	Démarrage du serveur	5
2	Fichiers LDIF et données de l'annuaire	6
2.1	Création de l'annuaire	6
2.2	Mise à jour des données	8
2.3	Recherche dans l'annuaire	10
3	Modification de schéma	11
3.1	Schéma <i>organizationalUnit</i>	12
3.2	Schéma <i>beerObject</i>	12
4	Accès via l'API <i>foo</i>	14
4.1	Perl	14
4.2	PHP	15
5	Pour aller plus loin	16

Bibliographie

- [1] Gerald CARTER. *LDAP, Administration système*. O'REILLY, édition française édition, 2003. ISBN : 2-84177-293-4.

Webographie

- [2] Harald Iana. Iana-registered private enterprise.
<http://www.alvestrand.no/objectid>.

organizationalUnit	
Description	The Organizational Unit object class is used to define entries representing subdivisions (folders) of organizations, such as Users, Management and others.
Object identifier	2.5.6.5
Type of object class	Structural class
Superior class	top
Mandatory attributes	ou
Optional attributes	userPassword searchGuide seeAlso businessCategory x121Address registeredAddress destinationIndicator preferredDeliveryMethod telexNumber teletexTerminalIdentifier telephoneNumber internationaliSDNNumber facsimileTelephoneNumber street postOfficeBox postalCode postalAddress physicalDeliveryOfficeName st description

TAB. 2 – ObjectClass : organizationalUnit