FORMATION LDAP

Julien Baudry
julien.baudry@univ-lehavre.fr
Ingénieur de Recherche
LITIS – Université du Havre

Table des matières

Conventions utilisées dans ce document

Les conventions syntaxiques utilisées dans ce document sont les suivantes :

```
Ceci est le contenu d'un fichier
```

```
# Ceci est une commande exécutée en tant que root (#)
```

```
$ Ceci est une commande exécutée en tant qu'utilisateur standard ($)
```

> *Ceci est une note*

Ceci est un texte standard

1 OpenLDAP

Exercice : Le chapitre suivant constitue un exercice à part entière et peut être effectué en même temps que la lecture du document, qui décrit toutes les étapes à suivre.

1.1 Introduction et bref historique

Après avoir étudié les nombreux concepts liés aux annuaires LDAP, passons désormais à la pratique et étudions l'implémentation libre la plus utilisée : OpenLDAP.

OpenLDAP est un projet libre diffusé sous licence "OpenLDAP Public License" (**http://www.openldap.org/license.html**). Il est supporté par la fondation OpenLDAP, créée en 1998 par une société du nom de "Net Boolean", fournisseur de services professionnels liés à la messagerie.

La première version d'OpenLDAP (1.0) sort en août 1998. Il faudra attendre août 2000 pour que la version 2.0 ne voie le jour, offrant le support de LDAP v3. La dernière version disponible est la version 2.4.7.

Liens :

*Site du projet : **http://www.openldap.org***
*Historique des versions : **http://www.openldap.org/software/roadmap.html***
*Historique du projet : **http://www.openldap.org/conf/odd-sfo-2003/keynote.html***

1.2 Installation

Comme pour tout logiciel, il est possible d'installer OpenLDAP par le biais de paquets binaires fournis par votre distribution, ou bien en compilant les sources. L'installation par paquets est souvent conseillée car elle facilite la maintenance du logiciel par la suite.

Les outils clients sont souvent dissociés des outils serveur et fournis dans des paquets séparés. Nous allons donc installer deux paquets, puisque je vous propose d'installer les clients sur la même machine que la machine serveur. Ceci n'est pas forcément le cas dans un environnement de production où les clients agirons à distance depuis une autres machine.

Sur une distribution de type Debian (Debian, Ubuntu, ...), les paquets à installer sont les suivants : **slapd** et **ldap-utils**. Saisissez donc, en tant que root, la commande suivante :

```
# apt-get install slapd ldap-utils
```

Une fenêtre apparaît alors et vous demande le mot de passe associé à l'annuaire que vous mettez en place. Indiquez ce que vous souhaitez, nous le changerons par la suite.

1.3 Les outils fournis par OpenLDAP

Le projet OpenLDAP implémente un serveur LDAP, mais également les commandes clientes permettant de manipuler des informations contenues dans l'annuaire.

1.3.1 Les commandes liées au serveur

Le paquet slapd fournit les binaires suivants :

```
# dpkg -L slapd | grep bin
/usr/sbin
/usr/sbin/slapd
/usr/sbin/slurpd
/usr/sbin/slaptest
/usr/sbin/slappasswd
/usr/sbin/slapindex
/usr/sbin/slapdn
/usr/sbin/slapcat
/usr/sbin/slapauth
/usr/sbin/slapadd
/usr/sbin/slapacl
```

Chacune de ces commandes permet d'agir directement au niveau du serveur OpenLDAP, notamment au niveau de sa base de données. Il est donc impératif de les exécuter sur le serveur où fonctionne le serveur OpenLDAP.

Démons :
slapd : le démon OpenLDAP !

slurpd : le démon de réplication

Commandes de manipulation de la base (backend) gérée par OpenLDAP
slapindex : crée les index au sein de la base

slapcat : effectue un dump (une copie intégrale) de la base

slapadd : ajoute des entrées LDIF dans la base

slappasswd : utilitaire de conversion de mots de passe

Commandes de test/validation :
slaptest : teste la validité du fichier de configuration slapd.conf

slapdn : teste la conformité d'un DN donné en ligne de commande

slapauth :

slapacl :

1.3.2 Les commandes clientes

Le paquet ldap-utils fournit les commandes suivantes :

```
# dpkg -L ldap-utils | grep bin
/usr/bin
/usr/bin/ldapsearch
/usr/bin/ldapmodify
/usr/bin/ldapdelete
/usr/bin/ldapmodrdn
/usr/bin/ldappasswd
/usr/bin/ldapwhoami
/usr/bin/ldapcompare
/usr/bin/ldapadd
```

Chaque commande cliente utilise le protocole LDAP pour agir sur l'annuaire. Elles peuvent donc, cette fois-ci, être utilisées à distance. Elles agissent en tant que clients LDAP standard. Nous verrons qu'il existe d'autres clients, graphiques notamment.

ldapsearch : effectue une recherche au sein de l'annuaire

ldapadd : ajoute une entrée

ldapdelete : supprime une entrée

ldapmodify : modifie une entrée (ajoute/suppr. un attribut, ajoute/suppr. une entrée, ...)

ldapmodrdn : modifie le rdn d'une entrée (renomme une entrée)

ldappasswd : modifie le mot de passe d'une entrée LDAP

ldapwhoami : affiche avec quel utilisateur le binding a eu lieu

ldapcompare : permet de comparer l'attribut d'une entrée à une valeur spécifiée

1.4 Configuration du serveur

L'intégralité de la configuration du serveur OpenLDAP (le démon slapd) s'effectue en modifiant le fichier slapd.conf, situé dans le répertoire /etc/ldap.

Voici un exemple de configuration, ainsi que l'explication des directives :

```
#######################################################################
# Directives globales

# Inclusion des schemas
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema

# Verification de la conformite des objets avec les schemas
schemacheck     on

# Ou sera stocke le PID du demon
pidfile         /var/run/slapd/slapd.pid

# Liste des arguments passes au demarrage du serveur
argsfile        /var/run/slapd.args

# Niveau de log
loglevel        0

# Emplacement des modules
# Chargement du module BDB (Berkeley DB)
modulepath /usr/lib/ldap
moduleload back_bdb

#######################################################################
# Declaration des options pour le premier type de backend utilise : bdb
# Toutes les options s'y appliquent jusqu'a la prochaine directive
# backend
backend          bdb
checkpoint 512 30

#######################################################################
#backend   <autre>

#######################################################################
# Declaration des options de la premiere "base", c'est a dire de la
# premiere (et unique ici) arborescence geree par notre annuaire
# Toutes les options s'y appliquent jusqu'a la prochaine directive
# database
```

```
database        bdb


# La racine de notre arborescence
suffix          "dc=univ-lehavre,dc=fr"

# Le compte administrateur de notre arborescence et son mot de passe
rootdn           "cn=admin,dc=univ-lehavre,dc=fr"
rootpw           "secret"

# Ou sont stockes les fichiers BDBs de notre arborescence
directory       "/var/lib/ldap"

# Options d'index
index           objectClass eq

# Sauvegarde de l'heure a laquelle est modifiee une entree
lastmod         on

# ACLs de notre premiere arborescence :
# Une personne non authentifiee peut s'authentifier
# Une personne authentifiee peut modifier son propre mot de passe
# Les autres n'ont pas accès à l'attribut mot de passe
access to attrs=userPassword
        by anonymous auth
        by self write
        by * none

# Tout le monde peut lire l'annuaire
access to *
        by * read

#######################################################################
# Autre arborescence
#database  <autre>
#suffix          "dc=debian,dc=org"
#[...]
```

Utilisez votre éditeur préféré, puis sauvegardez votre configuration. Testez-là ensuite pour voir si aucune erreur n'a été commise :

```
# slaptest -f /etc/ldap/slapd.conf
config file testing succeeded
```

Enfin, (re)-démarrez le serveur LDAP :

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

Le fichier de configuration est subdivisé en trois sections importantes :

la section globale (début du fichier)

la section concernant les options de backends (débute par "backend")

la section concernant les déclarations et les options des arborescences gérées (débute par "database")

Nous allons évoquer les directives les plus importantes ; n'hésitez pas à vous reporter au manuel du fichier de configuration pour plus d'informations :

```
# man 5 slapd.conf
```

1.4.1 L'inclusion des schémas

L'inclusion des schémas est effectuée par la directive "include". Comme nous l'avons étudié, les schémas LDAP permettent de définir les types de données contenus dans l'annuaire.

C'est grâce à ces inclusions au sein du fichier de configuration que l'on porte à la connaissance du serveur ces nouveaux types de données. Une fois les schémas chargés, il sera possible d'ajouter des entrées y faisant référence dans notre annuaire.

Plusieurs schémas sont fournis par défaut, je vous invite à regarder dans le répertoire /etc/ldap/schema pour les découvrir.

> *Note : la directive include inclut en fait un fichier de configuration (de manière générale). Il est donc possible de disposer de plusieurs fichiers de configuration spécifiques et de les regrouper de cette manière.*

1.4.2 Les niveaux de log

Il peut être important de savoir ce que fait exactement l'annuaire, ce, à des fins de débuggage par exemple. Pour ceci, nous avons à notre disposition la possibilité de changer le niveau de log dans le fichier slapd.conf.

Les niveaux de log disponibles sont les suivants (issus de "man slapd.conf") :

Valeur	Fonction correspondante
1	Appels de fonctions

Valeur	Fonction correspondante
2	Gestion des packets
4	Trace détaillée
8	Gestion des connexions
16	Affichage des packets envoyés et reçus
32	Gestion des filtres de recherche
64	Gestion du fichier de configuration
128	Gestion des ACLs
256	Affichage des connexions, opérations et résultats
512	Affichage des entrées retournées
1024	Affichage des communications avec les backends
2048	Parsing des entrées

Ces niveaux sont cumulables, c'est à dire qu'un niveau 48 équivaut aux niveaux 16 et 32. Un niveau 0 équivaut à un log désactivé.

Les logs sont gérés par syslog, ce qui signifie que vous pourrez consulter les informations logguées dans le fichier /var/log/syslog.

1.4.3 Les backends

Différents backends sont disponibles. Les plus couramment utilisés sont BDB (Berkeley DB, cf. : **http://www.sleepycat.com**) et LDBM. Ces deux backends sont des bases de données stockées dans des fichiers.

BDB est recommandé car réputé plus robuste que LDBM.

N'hésitez pas à consulter les pages de man de slapd-bdb et slapd-ldbm pour plus d'informations.

D'autres backends existent et permettent par exemple de stocker les informations dans de véritables SGBD, mais nous n'allons pas présenter ces fonctionnalités ici.

1.4.4 Les databases

Une section de database représente la déclaration d'une arborescence. Ceci implique plusieurs paramètres, dont une racine, un compte administrateur, ...

Voici les paramètres importants dans cette section :

La racine

Elle est spécifiée par la directive "suffix". La racine correspond souvent au **FQDN** (Fully Qualified Domain Name) de la machine associé aux attributs "dc".

```
suffix          "dc=univ-lehavre,dc=fr"
```

L'accès administrateur

Il est possible de déclarer un compte qui ne sera sujet à aucune limitation. Il s'agit en quelques sortes d'un compte "root". Ce compte peut correspondre ou non à une entrée dans l'annuaire. Il sera purement virtuel si aucun DN n'est effectivement stocké dans l'annuaire.

Ce compte particulier n'est pas soumis aux restrictions imposées par les ACLs (voir ci-dessous). Il est déclaré par la directive "rootdn". Son mot de passe est spécifié par la directive "rootpw".

> *Note : attention, ne confondez pas "rootdn" et DN racine, qui correspond à la base de notre annuaire ! Ici le "rootdn" est bien le dn d'un utilisateur ayant les droits root...*

Le mot de passe du rootdn peut être soit en clair, comme dans notre exemple, soit un hash généré par la commande slappasswd. Nous pouvons générer ce hash de cette manière :

```
# slappasswd -s "secret"
{SSHA}HPcdWtb5XII0Yt5Ly8QnxlV8bIyFk3WR
```

La valeur affichée est alors à copier-coller dans la valeur de la directive "rootpw" :

```
rootpw          "{SSHA}HPcdWtb5XII0Yt5Ly8QnxlV8bIyFk3WR"
```

Ceci permet de ne pas stocker en clair le mot de passe dans le fichier de configuration !

Les index

Les index sont un moyen d'accélérer les recherches au sein de l'annuaire. Dans le fichier de

configuration, il convient de préciser quels attributs seront le plus fréquemment utilisés pour les recherches et doivent donc être indexés. Ceci se fait par la directive "index" :

```
index           objectClass eq
```

Ici, nous activons la gestion des index sur les objectClass, ce qui semble un minimum !

Chaque index est destiné à faciliter un type de recherche. Le type d'index à créer est ici "eq", ce qui signifie qu'il sera efficace pour une recherche faisant intervenir une égalité stricte de chaînes. Voici une liste des types d'index disponibles et leur type de recherche associé :

Index	**Type de recherche (filtre), exemple**
eq	'uid=martymac', égalité stricte, pas d'utilisation de "wildcard" * (cf. sub)
sub	'uid=marty*', utilisation d'un wildcard
subinitial	optimisation de sub pour 'uid=marty*', wildcard à la fin
subfinal	optimisation de sub pour 'uid=*mac', wildcard au début
subany	optimisation de sub pour 'uid=*rtym*', wilcard au début ou à la fin
approx	'uid~=martymac', recherche par approximation sonore
pres	'objectclass=posixAccount', recherche de présence

Le type de recherche effectué est déduit du filtre passé au client qui effectue cette recherche.

Un ou plusieurs types de recherches peuvent être spécifiés pour un ou plusieurs attributs à la fois. Dans ce cas, la virgule sépare les différents éléments. Exemple :

```
index           uid,gecos,description eq,subinitial
index           uidNumber,gidNumber eq
```

Les index doivent être générés par l'administrateur pour être fonctionnels (commande "slapindex"), nous aborderons ce point par la suite.

Les listes d'accès (ACLs)

Les ALCs permettent de définir finement les droits d'accès à l'annuaire. La syntaxe générale des ACLs est la suivante :

```
access to <quoi> [ by <qui> <acces> [ <contrôle> ] ]+
```

Nous avons créé dans notre exemple deux ACLs :

```
access to attrs=userPassword
        by anonymous auth
        by self write
        by * none

access to *
        by * read
```

La première concerne l'attribut userPassword :

on autorise l'accès aux personnes non authentifiées uniquement pour une authentification (by anonymous auth)

on autorise une personne authentifiée à modifier son propre mot de passe (by self write)

enfin, on refuse l'accès à cet attribut aux autres personnes

La seconde ACL concerne toutes les informations contenues dans l'annuaire (*) :

on autorise tout le monde à les lire

> *Note :* *les ACls sont évaluées dans leur ordre d'apparition dans le fichier de configuration. OpenLDAP arrête leur évaluation lorsqu'il a trouvé une ACL faisant intervenir la cible recherchée. Les directives les plus générales doivent donc être situées* ***après*** *les directives s'appliquant à une cible particulière. C'est le cas ici avec l'ACL ciblant l'attribut userPassword, située avant celle ciblant toute information (*).*

Nous n'allons pas étudier ici plus en détail les ACLs, je vous invite à consulter la page de man de "slapd.access" pour plus de détails.

1.5 Administration du serveur

1.5.1 Introduction

Notre serveur est désromais configuré. Nous allons maintenant voir comment nous pouvons l'administrer.

> *Attention !!! Les commandes que nous utilisons ici n'utilisent pas le protocole LDAP mais accèdent directement à la base de données sous-jacente (BDB dans notre cas). Il est donc **impératif** de toujours couper le serveur LDAP avant d'utiliser une commande slap(...), afin d'éviter un accès concurrent depuis le serveur lui-même, ce qui pourrait corrompre la base de données.*

Coupez donc le serveur LDAP avant de continuer :

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

1.5.2 Slapindex

Nous avons configuré notre serveur pour qu'il utilise des index ; la première chose à effectuer avant d'utiliser notre serveur est donc de les générer. Il faut en effet initialiser les index pour qu'OpenLDAP puisse ensuite les utiliser et les maintenir.

L'opération de génération n'est à effectuer qu'une seule fois et ceci se fait par le biais de la commande slapindex.

```
# slapindex
```

1.5.3 Slapcat

Slapcat est une commande très utile au quotidien. Elle effectue un "dump" de la base LDAP au format LDIF. Il est conseillé de l'utiliser régulièrement pour effectuer des sauvegardes de notre annuaire.

Par défaut, slapcat affiche les informations sur la sortie standard, il faut donc la rediriger vers un fichier pour obtenir notre sauvegarde :

```
# slapcat > sauvegarde.ldif
```

1.5.4 Slapadd

Slapadd est l'inverse de slapcat. Cette commande permet de peupler notre annuaire en utilisant un fichier LDIF. Elle est typiquement utilisée pour restaurer une sauvegarde effectuée avec slapcat :

```
# slapadd < sauvegarde.ldif
```

1.5.5 Arrêt et démarrage du serveur

L'arrêt et le démarrage du serveur LDAP se font par le biais du script /etc/init.d/slapd :

```
/etc/init.d/slapd [start|stop|restart]
```

L'administration du serveur étant terminée, je vous propose de le démarrer et de découvrir les commandes clientes fournies par OpenLDAP :

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

1.6 Utilisation des outils clients

1.6.1 Introduction

Nous avons étudié les outils d'administration du serveur : les outils slap(...), nous allons maintenant étudier les outils clients. A la différence des outils slap(...), les outils ldap(...) utilisent le protocole LDAP, il peuvent donc être mis en oeuvre depuis n'importe quelle machine disposant d'un accès réseau au serveur LDAP. Ils utilisent bien évidemment le format LDIF pour échanger des informations avec le serveur.

Nous allons étudier les différentes possibilités offertes par ces outils à travers des exemples simples.

- Executez les différentes commandes décrites ci dessous (ldapadd, ldapdelete, ldapsearch).
- A l'aide d'un client LDAP graphique validez les modifications engendrées sur votre annuaire.

1.6.2 Ajouter une entrée : ldapadd

Pour ajouter une entrée dans l'annuaire, il faut utiliser la commande ldapadd. Sa syntaxe est la suivante :

```
ldapadd -W -D <binddn> -x -H ldap://<serveur> -f <fichier.ldif>
```

L'option "-W" active la demande de mot de passe pour s'authentifier en tant que <binddn>. L'option "-x" permet de ne pas utiliser SASL pour l'authentification. Enfin, le fichier LDIF source doit contenir une (ou plusieurs) entrée(s) à insérer et l'intégralité de ses (leurs) attributs.

Exemple :

Fichier LDIF à insérer (fichier.ldif) :

```
dn: uid=dupont,ou=users,ou=iut,dc=univ-lehavre,dc=fr
```

```
objectClass: account
objectClass: posixAccount
cn: dupont
uid: dupont
uidNumber: 10001
gidNumber: 1024
homeDirectory: /home/dupont
userPassword:: e0NSWVBUfXZKblR0TjVSaXQ0Tmc=
loginShell: /bin/sh
gecos: dupont
description: dupont
```

Insertion de l'entrée :

```
# ldapadd -W -D "cn=admin,dc=univ-lehavre,dc=fr" -x -H
ldap://localhost -f fichier.ldif
Enter LDAP Password:
adding new entry "uid=dupont,ou=users,ou=iut,dc=univ-lehavre,dc=fr"
```

Nous insérons cette entrée depuis le serveur lui-même (localhost) et nous utilisons ici le compte "admin" déclaré dans le fichier de configuration. Notez que cette commande ne fonctionnera pas si vous n'avez pas initialisé l'annuaire avec les entrées de base que nous avons vues auparavant : ou=users, ou=iut et dc=univ-lehavre,dc=fr. Voyons comment nous pouvons créer ces entrées basiques.

1.6.3 Initialiser l'annuaire

L'initialisation de l'annuaire n'est qu'un ajout massif de plusieurs entrées. Cet ajout massif peut se faire par le biais de slapadd si vous possédez déjà un dump de l'annuaire et si vous vous situez sur le serveur.

A distance, c'est l'outil ldapadd qui va nous permettre d'effectuer cette opération. Il suffit de fournir à ldapadd un fichier LDIF contenant plusieurs entrées qui seront ajoutées dans le même ordre avec lequel elles apparaissent dans le fichier.

Ce fichier va donc tout d'abord contenir l'entrée de la racine, qui est nécessaire, puis chacune des "ou" que nous avons vues en exemple. Enfin, les feuilles seront constituées d'utilisateurs et de groupes.

Fichier LDIF à insérer (fichier.ldif) :

```
dn: dc=univ-lehavre,dc=fr
```

```
objectClass: dcObject
objectClass: organization
dc: univ-lehavre
o: univ-lehavre
description: univ-lehavre

dn: ou=iut,dc=univ-lehavre,dc=fr
objectClass: top
objectClass: organizationalUnit
ou: iut

dn: ou=users,ou=iut,dc=univ-lehavre,dc=fr
objectClass: top
objectClass: organizationalUnit
ou: users

dn: ou=groups,ou=iut,dc=univ-lehavre,dc=fr
objectClass: top
objectClass: organizationalUnit
ou: groups

dn: cn=etudiants,ou=groups,ou=iut,dc=univ-lehavre,dc=fr
objectClass: posixGroup
cn: etudiants
gidNumber: 2000

dn: uid=dupont,ou=users,ou=iut,dc=univ-lehavre,dc=fr
objectClass: account
objectClass: posixAccount
cn: dupont
uid: dupont
uidNumber: 10001
gidNumber: 2000
homeDirectory: /home/dupont
userPassword:: e0NSWVBUfWRhSDJadHI4dElnZFE=
loginShell: /bin/sh
gecos: dupont
description: dupont

dn: uid=durand,ou=users,ou=iut,dc=univ-lehavre,dc=fr
objectClass: account
objectClass: posixAccount
cn: durand
uid: durand
uidNumber: 10002
gidNumber: 2000
homeDirectory: /home/durand
userPassword:: e0NSWVBUfTQzZXltaHBSUzBqQVk=
loginShell: /bin/sh
gecos: durand
description: durand
```

Le fichier ci-dessus comprend :

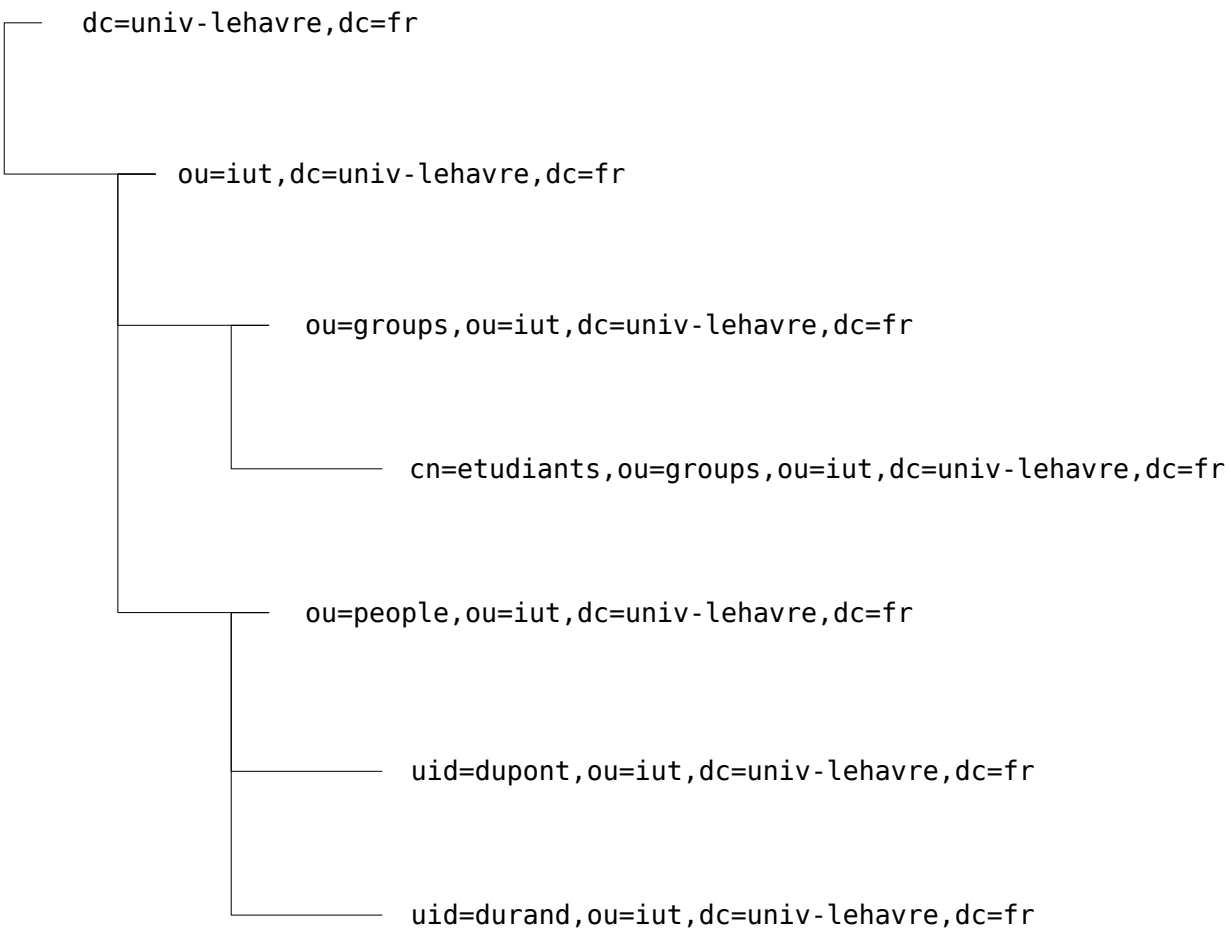
l'entrée de la racine (indispensable)

l'entrée des deux ou : users et groups

un groupe : utilisateurs

deux utilisateurs : dupont et durand appartenant au groupe etudiant (attr. gidNumber)

Voici un schéma représentant cette arborescence :



Insertion de cette arborescence :

```
# ldapadd -W -D "cn=admin,dc=univ-lehavre,dc=fr" -x -H
ldap://localhost -f fichier.ldif
```

1.6.4 Rechercher une entrée : ldapsearch

La commande ldapsearch permet d'effectuer une recherche au sein de l'annuaire. Voici sa syntaxe :

```
ldapsearch -x -H ldap://<serveur> -b <base> [-s portée] [filtre] [attributs]
```

Elle reprend bien évidemment les concepts que nous avons abordés jusqu'ici :

- la base de la recherche
- la portée de la recherche (base, one ou sub) - sub est la portée par défaut
- le filtre
- le ou les attributs que l'on souhaite afficher - l'entrée entière est affichée par défaut

Il est possible de s'authentifier, si nécessaire, avec l'option -D (et l'option -W), mais notre annuaire est ici configuré pour permettre l'accès en lecture à tout le monde.

Exemples :

On recherche tous les uid commençant par "dupon*" à partir de la racine de l'annuaire :

```
ldapsearch -x -H ldap://localhost -b "dc=univ-lehavre,dc=fr"
"(uid=dupon*)"
```

On recherche toutes les entrées ayant un gidNumber égal à 2000 :

```
ldapsearch -x -H ldap://localhost -b dc=univ-lehavre,dc=fr
"(gidNumber=2000)"
```

Cette commande nous retourne les 2 utilisateurs, mais aussi le groupe car il possède lui aussi l'attribut gidNumber. Améliorons notre requête pour ne retourner que les deux comptes utilisateurs :

```
ldapsearch -x -H ldap://localhost -b dc=univ-lehavre,dc=fr
"(&(gidNumber=2000)(objectClass=posixAccount))"
```

On affiche enfin uniquement leur répertoire home (et pas la totalité de l'entrée comme c'est le cas par défaut) :

```
ldapsearch -x -H ldap://localhost -b dc=univ-lehavre,dc=fr
"(&(gidNumber=2000)(objectClass=posixAccount))" homeDirectory
```

Le résultat de cette dernière requête est le suivant :

```
# [...]
# dupont, users, iut, univ-lehavre.fr
dn: uid=dupont,ou=users,ou=iut,dc=univ-lehavre,dc=fr
homeDirectory: /home/dupont

# durand, users, iut, univ-lehavre.fr
dn: uid=durand,ou=users,ou=iut,dc=univ-lehavre,dc=fr
homeDirectory: /home/durand
# [...]
```

1.6.5 Consultation de l'entrée rootDSEDirectory

L'accès à cette entrée spécifique se fait en interrogeant l'annuaire avec un suffixe vide (c'est-à-dire par conséquent au niveau même de la racine) et en spécifiant une profondeur (scope) à la valeur "base".
En utilisant la commande ldapsearch, la synthaxe de la requête est :

```
ldapsearch -L -x -b "" -H ldap://<serveur>:<port>/ -s base "" +
```

Exemple

```
# ldapsearch -L -x -b "" -H ldap://localhost -s base "" +
```

1.6.6 Consultation de l'entrée subschema

Le schéma de la base d'annuaire est stocké dans une entrée particulière (cn=subschema).

Lorsqu'on utilise la requête ldapsearch, les deux requêtes précédentes deviennent respectivement :

```
# ldapsearch -L -x -H ldap://hostname:port/ -b "cn=subschema"
```

et

```
# ldapsearch -L -x -H ldap://hostname:port/ -b "cn=subschema" +
```

1.6.7 Supprimer une entrée : ldapdelete

La suppression d'une entrée se fait par la commande ldapdelete. Voici sa syntaxe :

```
ldapdelete -W -D <binddn> -x -H ldap://<serveur> <dn>
```

Puisqu'un effacement correspond à une écriture, il faudra, la plupart du temps, s'authentifier (à la

différence de ldapsearch).

Il est possible d'effacer récursivement une branche complète en utilisant l'option "-r" sur le noeud de la branche. Attention, commande potentiellement dangereuse !

> *Note : Il est possible de réinitialiser un annuaire par la méthode dite "sauvage et brutale, mais simple et rapide" ! En effet, il est possible de simplement supprimer les fichiers de la base de données de l'annuaire et de le redémarrer. Ces fichiers sont souvent situés dans le répertoire /var/lib/ldap (cf. directive "directory" du fichier de configuration). Attention si vous possédez plusieurs bases à ne pas toutes les effacer...*

Exemples :

Suppression de l'utilisateur dupont :

```
#ldapdelete -x -H ldap://localhost -W -D "cn=admin,dc=univ-
lehavre,dc=fr" "uid=odie,ou=users,dc=univ-lehavre,dc=fr"
```

Suppression de la branche users :

```
#ldapdelete -x -H ldap://localhost -W -D "cn=admin,dc=univ-
lehavre,dc=fr" -r "ou=users,dc=univ-lehavre,dc=fr"
```

- Re-créez cette branche ainsi que ses deux utilisateurs
- Ajouter deux nouvelle branches sous la racine : isel et fac (avec leur arborescence people, groups)

1.6.8 Modifier une entrée : ldapmodify

La commande ldapmodify est un peu le "couteau suisse" des annuaires LDAP ! Elle va permettre d'effectuer toutes sortes d'opérations, y compris l'ajout et la suppression d'entrées. Sa syntaxe est la suivante :

```
ldapmodify -W -D <binddn> -x -H ldap://<serveur> -f <fichier.ldif>
```

ldapmodify peut se substituer à ldapadd et ldapdelete, mais vous allez voir que son utilisation n'est

pas des plus simples ! En effet, tout passe par le fichier ldif indiqué en entrée et qui va décrire l'opération à effectuer...

Voici une liste (non exhaustive) d'opérations possibles :

- ajouter d'une entrée
- supprimer d'une entrée
- ajouter un attribut
- supprimer un attribut
- modifier un attribut

> *Note :* *N'hésitez pas à consulter les pages de man de "slapd.replog" et "ldif" pour une liste exhaustive des opérations que l'on peut effectuer. La syntaxe est la même syntaxe que celle employée dans les mécanismes de réplications que nous avons évoqués plus haut.*

Ajouter une entrée :

Ajoutons un utilisateur "john".

Fichier LDIF (fichier.ldif) :

```
dn: uid=john,ou=users,ou=iut,dc=univ-lehavre,dc=fr
changetype: add
objectClass: account
objectClass: posixAccount
cn: john
uid: john
uidNumber: 10003
gidNumber: 2000
homeDirectory: /home/john
userPassword:: e0NSWVBUfTg0QmNhL1BhL2tIUC4=
loginShell: /bin/sh
gecos: john
description: john
```

On remarque la présence d'un attribut "changetype" en plus de chacun des attributs de l'entrée que nous souhaitons ajouter.

Application de la modification :

```
# ldapmodify -W -D "cn=admin,dc=univ-lehavre,dc=fr" -x -H
```

```
ldap://localhost -f fichier.ldif
```

Supprimer une entrée :

Le principe est le même que pour l'ajout d'une entrée. Cette fois, la valeur de "changetype" n'est plus "add" mais "delete".

Fichier LDIF (fichier.ldif) :

```
dn: uid=john,ou=usersou=iut,dc=univ-lehavre,dc=fr
changetype: delete
```

Application de la modification : cf. point précédent, il s'agit de la même commande !

Ajouter un attribut :

L'ajout d'un attribut s'effectue par le changetype "modify" et par un nouvel attribut "add" qui précise quel attribut ajouter. Ici, nous allons ajouter une seconde description pour l'utilisateur "dupont". L'attribut "description" devient donc multi-valué.

```
dn: uid=dupont,ou=users,ou=iut,dc=univ-lehavre,dc=fr
changetype: modify
add: description
description: etudiant ERASMUS
```

Après la modification, l'utilisateur "dupont" possède les attributs suivants :

```
ldapsearch -x -H ldap://localhost -b "ou=users,ou=iut,dc=univ-
lehavre,dc=fr" "(uid=dupont)"

#[...]
# dupont, users, iut, univ-lehavre.fr
dn: uid=dupont,ou=users,ou=iut,dc=univ-lehavre,dc=fr
objectClass: account
objectClass: posixAccount
cn: dupont
uid: dupont
uidNumber: 10001
gidNumber: 2000
homeDirectory: /home/dupont
loginShell: /bin/sh
gecos: dupont
description: dupont
description: etudiant ERASMUS
#[...]
```

Supprimer un attribut :

La suppression d'attribut s'effectue via le changetype "modify" et l'utilisation d'un nouvel attribut : "delete".

Supprimons la description, fort négative pour Dupont, que nous venons d'ajouter :

```
dn: uid=dupont,ou=iut,ou=users,dc=univ-lehavre,dc=fr
changetype: modify
delete: description
description: etudiant ERASMUS
```

Notez qu'il est possible de ne pas spécifier la valeur de la description à supprimer. Dans ce cas, toutes les descriptions seront supprimées.

Modifier un attribut :

Pour modifier un attribut, le "changetype" à employer est, ici encore, "modify". L'attribut supplémentaire à ajouter est l'attribut "replace" qui va préciser quel attribut remplacer. Enfin, nous spécifions la nouvelle valeur de l'attribut.

```
dn: uid=dupont,ou=iut,ou=users,dc=univ-lehavre,dc=fr
changetype: modify
replace: description
description: Etudiant ERASMUS (2007-2008)
```

Remarquez que la modification ci-dessus remplace toutes les descriptions par celle qui a été spécifiée. Il n'est pas possible de modifier uniquement l'une des valeurs d'un attribut multi-valué.

Il faudra ruser pour effectuer cette dernière opération et le faire en deux temps : d'abord supprimer l'attribut désiré, ensuite ajouter le nouvel attribut. Imaginons que nous soyons dans le cas où dupont ait deux attributs :

```
ldapsearch -x -H ldap://localhost -b "ou=users,ou=iut,dc=univ-
lehavre,dc=fr" "(uid=dupont)"

#[...]
# dupont, users, martymac.com
dn: uid=dupont,ou=users,ou=iut,dc=univ-lehavre,dc=fr
objectClass: account
objectClass: posixAccount
cn: dupont
```

```
uid: dupont
uidNumber: 10001
gidNumber: 2000
homeDirectory: /home/dupont
loginShell: /bin/sh
gecos: dupont
description: dupont
description: Etudiant ERASMUS
#[...]
```

Si nous souhaitons remplacer uniquement "Etudiant ERASMUS" par "Etudiant ERASMUS (Allemagne)", nous pouvons utiliser le fichier ci-dessous :

```
dn: uid=dupont,ou=users,ou=iut,dc=univ-lehavre,dc=fr
changetype: modify
delete: description
description: dupont

dn: uid=dupont,ou=users,dc=univ-lehavre,dc=fr
changetype: modify
add: description
description: Etudiant ERASMUS (Allemagne)
```

Ce qui peut s'écrire de manière plus concise par l'utilisation du tiret "-" qui permet de chaîner les actions pour un même DN :

```
dn: uid=dupont,ou=users,dc=univ-lehavre,dc=fr
changetype: modify
delete: description
description:  Etudiant ERASMUS
-
add: description
description: Etudiant ERASMUS (2007-2008)
```

1.6.9 Renommer une entrée : ldapmodrdn

L'outil ldapmodrdn permet de modifier le RDN (uniquement) d'une entrée. Il s'utilise de cette manière :

```
ldapadd -W -D <binddn> -x -H ldap://<serveur> <dn> <nouveau_rdn>
```

Exemple :

Pour renommer "dupont" en "martin", nous pourrions saisir cette commande :

```
ldapmodrdn -W -D "cn=admin,dc=univ-lehavre,dc=fr" -x -H
ldap://localhost "uid=dupont,ou=iut,ou=users,dc=univ-lehavre,dc=fr"
"uid="
```

L'attribut "uid: pookie" sera ajouté automatiquement à l'entrée car il compose le nouveau RDN. L'ancienne valeur de l'uid ("uid: dupont") sera conservée.

1.6.10 Configuration des outils clients

Voici une information qui vous simplifiera la tâche par la suite : il existe un fichier de configuration pour les outils clients ! Ce fichier contient les options que les commandes clientes doivent utiliser par défaut : l'adresse du serveur cible, le binddn, etc... Jusqu'ici, ces options étaient passées à chaque fois en lignes de commandes ; renseigner ce fichier de configuration évitera cette tâche répétitive et fastidieuse.

Il existe deux fichiers de configuration : **/etc/ldap/ldap.conf** et **~/.ldaprc**. Le premier est disponible pour tous les utilisateurs ; le second peut être défini par l'utilisateur et permet de surcharger les options spécifiées par le premier.

Voici un exemple de fichier /etc/ldap/ldap.conf :

```
# Configuration des outils clients (voir "man ldap.conf")

# Racine
BASE dc=univ-lehavre,dc=fr
# Nom et port de l'annuaire
URI  ldap://localhost:389
```

Ainsi, une interrogation de l'annuaire devient simplement :

```
# ldapsearch -x
```

1.7 Applications web (PHP) et LDAP

L'objectif de ce chapitre est de montrer comment intéragir avec un annuaire à l'aide d'un langage de script. Le langage de script utilisé sera PHP.

L'utilisation de ce type de langage permet plusieurs opérations :
- Administrer l'annuaire LDAP en bénéficiant de bibliothèques évoluées (modifications de masse)
- Proposer des interfaces graphiques conviviales pour les utilisateurs afin de maintenir à jour les données.

PHP permet la connexion et l'envoi de requêtes sur un annuaire LDAP.

Un serveur LDAP est conçu pour être capable de gérer les opérations suivantes :

- établir la connexion avec l'annuaire
- rechercher des entrées
- comparer des entrées
- ajouter des entrées
- modifier des entrées
- supprimer des entrées
- annuler ou abandonner une opération
- fermer la connexion avec l'annuaire

Ainsi PHP fournit un ensemble de fonctions (pour peu que le module LDAP soit installé) permettant de réaliser ces opérations. Ces fonctions seront explicitées au long de l'article.

L'interrogation d'un serveur LDAP avec PHP se fait selon une séquence simple, nécessitant un nombre peu élevé de fonctions spécialisées. La séquence basique est la suivante :
Etablissement de la connexion avec le serveur LDAP

- Liaison et authentification sur le serveur (appelé *bind* en anglais)
- Recherche d'une entrée (ou bien une autre opération)
- Exploitation des résultats (uniquement dans le cas d'une recherche)
- Fermeture de la connexion

En PHP cela donne :

```
ldap_connect()     // establish connection to server
   |
ldap_bind()        // anonymous or authenticated "login"
   |
do something like search or update the directory
and display the results
   |
ldap_close()       // "logout"
```

1.7.1 Connexion à un annuaire

Avant de pouvoir interroger le serveur LDAP, il est essentiel d'initier la connexion. Pour cela l'interpréteur PHP a besoin de connaître quelques renseignements relatifs à l'annuaire :

- L'adresse du serveur
- Le port sur lequel le serveur fonctionne
- La racine de l'annuaire
- Le login de l'utilisateur (généralement *root*) ainsi que son mot de passe

```
<?
        // Fichier de configuration pour l'interface PHP
        //  de notre annuaire LDAP
        $server = "localhost";
        $port   = "389";
        $racine = "dc=univ-lehavre,dc=fr";
        $rootdn = "cn=admin,dc=univ-lehavre,dc=fr";
        $rootpw = "secret";

        echo "Connexion...<br>";

        $ds=ldap_connect($server) or exit(">>Connexion au serveur LDAP
echoué<<") ;

        if ($ds) {
                // on s'authentifie en tant que super-utilisateur, ici,
ldap_admin
                $r=ldap_bind($ds,$rootdn,$rootpw);

                // Ici les opérations à effectuer
                echo "Déconnexion...<br>";

                // On ferme la connexion
                ldap_close($ds);
        }
        else {
                echo  "Impossible de se connecter au serveur LDAP";
        }

?>
```

1.7.2 Consultation d'entrées

1.7.3 Ajouter une entrée avec ldap_add()

La fonction *ldap_add()* permet d'ajouter des entrées à un annuaire LDAP auquel on s'est préalablement connecté (et lié). Voici sa syntaxe :

```
int ldap_add (int identifiant, string dn, array entry)
```

La fonction *ldap_add()* admet en paramètre l'identifiant du serveur LDAP retourné par la fonction *ldap_connect()* ainsi que le nom distingué de l'entrée (c'est-à-dire son emplacement dans l'annuaire) et un tableau contenant l'ensemble des valeurs des attributs de l'entrée.

Lorsqu'un attribut est multivalué (c'est-à-dire lorsqu'un attribut est lui-même composé de plusieurs valeurs), celles-ci sont indexées dans une case du tableau (les indices du tableau commencent à 0). Dans l'exemple ci-dessous par exemple, l'attribut "mail" possède plusieurs valeurs :

```
<?php
$ds=ldap_connect($server);  // On suppose que le serveur LDAP est sur cet
hote
if ($ds) {
        $r=ldap_bind($ds,$rootdn,$rootpw);
        // preparation des données
        $entry["cn"]="Baudry";
        $entry["sn"]="Julien";
        $entry["mail"][0]="julien.baudry@univ-lehavre.fr";
        $entry["mail"][1]="baudryj@univ-lehavre.fr";
        $entry["objectclass"]="person";
        // Ajout des données dans l'annuaire
        $r=ldap_add($ds, "cn=Julien Baudry, ou=iut, dc=univ-lehavre, dc=fr",
$entry);
        ldap_close($ds);
} else {
        echo "Connexion au serveur LDAP impossible";
}
?>
```

Comparer une entrée avec ldap_compare()

La fonction *ldap_compare()* permet de comparer la valeur d'un attribut d'une entrée de l'annuaire LDAP, auquel on s'est préalablement connecté (et lié), à une valeur passée en paramètre. Voici la syntaxe de la fonction *ldap_compare()* :

```
int ldap_compare (int identifiant, string dn, string attribut, string valeur)
```

La fonction *ldap_compare()* admet en paramètre l'identifiant du serveur LDAP retourné par la fonction *ldap_connect()*, le nom distingué de l'entrée (c'est-à-dire son emplacement dans l'annuaire) ainsi que le nom de l'attribut de l'entrée et la valeur à laquelle on veut comparer sa valeur dans l'annuaire.

En cas d'erreur, la fonction *ldap_compare()* renvoie la valeur -1, en cas de réussite elle renvoie *TRUE*, enfin dans le cas contraire elle renvoie *FALSE*.

L'exemple suivant montre comment comparer un mot de passe avec celui stocké dans l'annuaire pour un utilisateur donné :

```
<?php
$ds=ldap_connect($server);
if ($ds) {
        $r=ldap_bind($ds,$rootdn,$rootpw);
        // preparation des données
        $dn="cn=Julien Baudry, ou=iut,dc=univ-lehavre,dc=fr";
        $valeur="MonMot2Passe";
        $attribut="password";
        // Comparaison du mot de passe à celui dans l'annuaire
        $resultat=ldap_compare($ds, $dn, $attribut, $valeur);
        if ($resultat == -1) {
        echo "Erreur:".ldap_error($ds);
        }
        elseif ($resultat == TRUE) {
        echo "Le mot de passe est correct";
        }
        else ($resultat == FALSE) {
        echo "Le mot de passe est erronné...";
        }
        ldap_close($ds);
} else {
        echo "Connexion au serveur LDAP impossible";
}
?>
```

Notez l'utilisation de la fonction *ldap_error()* sur laquelle nous reviendrons ultérieurement pour afficher les détails de l'erreur !

1.7.4 Supprimer une entrée avec *ldap_delete()*

La fonction *ldap_delete()* permet de supprimer des entrées d'un annuaire LDAP. Voici sa syntaxe :

```
int ldap_delete (int identifiant, string dn)
```

La fonction *ldap_delete()* admet uniquement deux paramètres :

- l'identifiant du serveur LDAP retourné par la fonction *ldap_connect()*
- le nom distingué de l'entrée à supprimer.

Une fois de plus, cette fonction renvoie *TRUE* en cas de réussite, *FALSE* en cas d'échec.

L'exemple suivant illustre la suppression d'un élément de l'annuaire :

```
<?php
$ds=ldap_connect($server);  // On suppose que le serveur LDAP est sur cet
hote
if ($ds) {
        $r=ldap_bind($ds,$rootdn,$rootpw);
        // preparation des données
        $dn="cn=Julien Baudry,ou=iut,dc=univ-lehavre,dc=fr";
        // Supression de l'entrée de l'annuaire
        $r=ldap_delete($ds, $dn);
        ldap_close($ds);
} else {
        echo "Connexion au serveur LDAP impossible";
}
?>
```

1.7.5 Modifier une entrée avec *ldap_modify()*

La fonction *ldap_modify()* permet de modifier une entrée de l'annuaire LDAP. Sa syntaxe est la même que celle de la fonction *ldap_add()* :

```
#int ldap_modify (int identifiant, string dn, array entry)
```

La fonction *ldap_modify()* admet en paramètre l'identifiant du serveur LDAP retourné par la fonction *ldap_connect()* ainsi que le nom distingué de l'entrée (c'est-à-dire son emplacement dans l'annuaire) et un tableau contenant l'ensemble des valeurs des attributs de l'entrée à modifier.

Lorsqu'un attribut est multivalué (c'est-à-dire lorsqu'un attribut est lui-même composé de plusieurs valeurs), celles-ci sont indexées dans une case du tableau (les indices du tableau commencent à 0). Dans l'exemple ci-dessous par exemple, l'attribut "mail" possède plusieurs valeurs :

```
<?php
$ds=ldap_connect($server);  // On suppose que le serveur LDAP est sur cet
hote
if ($ds) {
        $r=ldap_bind($ds,$rootdn,$rootpw);
        // preparation des données
        $entry["cn"]="Baudry";
        $entry["sn"]="Julien";
        $entry["mail"][0]="julien.baudry@univ-lehavre.fr";
        $entry["mail"][1]="baudryj@univ-lehavre.fr";
        $entry["objectclass"]="person";
        // Ajout des données dans l'annuaire
        $r=ldap_modify($ds, "cn=Julien Baudry,ou=iut,dc=univ-lehavre,dc=fr",
$entry);
        ldap_close($ds);
} else {
        echo "Connexion au serveur LDAP impossible";
}
?>
```

1.7.6 Rechercher une entrée avec *ldap_search()*

La recherche d'entrée dans l'annuaire est sans aucun doute la fonction la plus utile parmi les fonctions LDAP de PHP¨car un annuaire est prévu pour être plus souvent sollicité en lecture (recherche) qu'en écriture (ajout/suppression/modification).

La fonction *ldap_search()* permet de rechercher une ou plusieurs entrées de l'annuaire LDAP à l'aide du DN de base, c'est-à-dire le niveau de l'annuaire à partir duquel la recherche est effectuée, ainsi qu'un filtre représentant le type de recherche que l'on désire effectuer. Sa syntaxe est la suivante :

```
int ldap_search (int identifiant, string base_dn,
        string filter [, array attributs [, int attrsonly [,
        int sizelimit [, int timelimit [, int deref]]]]])
```

La fonction *ldap_search()* admet en paramètre l'identifiant du serveur LDAP retourné par la fonction *ldap_connect()* ainsi que le nom distingué du dossier de base (c'est-à-dire celui à partir duquel la recherche doit s'effectuer) et le filtre de la recherche. La fonction *ldap_search()* est par défaut configurée avec l'option de récursivité *LDAP_SCOPE_SUBTREE* ce qui signifie que la recherche se fait dans toutes les branches filles du dossier de base.

Le paramètre *attributs* permet de restreindre les attributs et les valeurs retournées, c'est-à-dire qu'il s'agit d'un tableau contenant le nom des attributs (chaînes de caractères) des attributs que l'on désire utiliser. Par défaut l'intégralité des attributs des entrées est renvoyée par le serveur, ce qui peut donner un nombre de données très important.

Le paramètre *attrsonly* permet de demander à l'annuaire de retourner uniquement les types d'attributs et non leurs valeurs lorsqu'il vaut 1. Par défaut (ou lorsque ce paramètre vaut 0) les types des attributs **ainsi** que leurs valeurs sont retournés par le serveur.

Le sixième paramètre *sizelimit* comme son nom l'indique permet de limiter le nombre maximum de résultat retourné par l'annuaire afin de réduire le volume des données retournées. Il faut noter que si le serveur est configuré pour retourner moins de résultats, une valeur supérieure de l'attribut ne permettra pas de dépasser la valeur inscrite dans la configuration du serveur. La valeur *0* indique qu'aucune limite autre que celle imposée par le serveur n'est définie.

Le septième paramètre *timelimit* permet de limiter le temps maximal de la recherche pris par le serveur. Il faut noter que si le serveur est configuré pour retourner moins de résultats, une valeur supérieure de l'attribut ne permettra pas de dépasser la valeur inscrite dans la configuration du serveur. La valeur *0* indique qu'aucune limite autre que celle imposée par le serveur n'est définie.

Enfin le huitième paramètre *deref* permet d'indiquer selon sa valeur la façon de procéder avec les alias lors de la recherche. Les valeurs possibles de ce paramètre sont les suivantes :

- *LDAP_DEREF_NEVER* : les alias ne sont jamais déréférencés. Il s'agit de la valeur par

défaut.

- *LDAP_DEREF_SEARCHING* : les alias sont déréférencés uniquement pendant la recherche et non pendant leur localisation.
- *LDAP_DEREF_FINDING* : les alias sont déréférencés uniquement pendant leur localisation et non lors de la recherche.
- *LDAP_DEREF_ALWAYS* : les alias sont toujours déréférencés.

L'exemple suivant permet de connaître le nombre de résultats retournés pour une recherche d'une personne dont le nom ou le prenom commence par la chaîne *$person* passée en paramètre :

```
<?php
$ds=ldap_connect($server);  // On suppose que le serveur LDAP est sur cet
hote
if ($ds) {
        $r=ldap_bind($ds,$rootdn,$rootpw);
        $dn = "dc=univ-lehavre,dc=fr";
        $filtre="(|(sn=$person*)(cn=$person*))";
        $restriction = array( "cn", "sn", "mail");
        $sr=ldap_search($ds, $dn, $filtre, $restriction);
        $info = ldap_get_entries($ds, $sr);
        print $info["count"]." enregistrements trouves
";
        ldap_close($ds);
} else {
        echo "Connexion au serveur LDAP impossible";
}
?>
```

Toutefois, une fois l'opération de recherche effectuée, il s'agit d'exploiter les résultats obtenus. Ainsi, la majeure partie des fonctions LDAP ont pour but le traitement des résultats de la recherche. Dans l'exemple ci-dessus, la fonction *ldap_get_entries()* permet de récupérer des informations sur les entrées retournées par la fonction *ldap_search()*.

1.7.7 Traitement des résultats

De nombreuses fonctions LDAP permettent d'exploiter les résultats renvoyés par la fonction *ldap_search()*. Ces fonctions ont un nom commençant généralement par *ldap_get_* suivi du nom de l'élément à récupérer :

- *ldap_get_dn()* permet de récupérer le DN de l'entrée
- *ldap_get_entries()* permet de récupérer l'ensemble des entrées
- *ldap_get_option()* permet de récupérer la valeur d'une option
- *ldap_get_values()* permet de récupérer toutes les valeurs d'une entrée
- *ldap_get_values_len()* permet de récupérer les valeurs binaires d'une entrée
- *ldap_count_entries()* permet de récupérer le nombre d'entrées retournées par la fonction de recherche

Récupérer le nombre d'entrées retournées

La fonction *ldap_count_entries()* permet de connaître le nombre d'entrées retournées par la fonction *ldap_search()*. Sa syntaxe est la suivante :

```
#int ldap_count_entries (int link_identifier, int result_identifier)
```

La fonction *ldap_count_entries()* admet en paramètre l'identifiant du serveur LDAP retourné par la fonction *ldap_connect()* ainsi que l'identifiant du résultat retourné par la fonction *ldap_search()* et retourne un entier représentant le nombre d'entrées stockées dans le résultat de la recherche.

Voici un exemple d'utilisation :

```
<?php
$ds=ldap_connect($server);  // On suppose que le serveur LDAP est sur cet
hote
if ($ds) {
        $r=ldap_bind($ds,$rootdn,$rootpw);
        $dn = "dc=univ-lehavre,dc=fr";
        $filtre="(|(sn=$person*)(cn=$person*))";
        $restriction = array( "cn", "sn", "mail");
        $sr=ldap_search($ds, $dn, $filtre, $restriction);
        $nombre = ldap_count_entries($ds, $sr);
        print $nombre." enregistrements trouves
";
        ldap_close($ds);
} else {
        echo "Connexion au serveur LDAP impossible";
}
```

?>

1.7.8 Récupérer les entrées retournées

La fonction *ldap_get_entries()* permet de récupérer l'ensemble des entrées retournées par la fonction *ldap_search()* ainsi que de lire les attributs associés et leur(s) valeur(s). Sa syntaxe est la suivante :

```
array ldap_get_entries (int link_identifier, int result_identifier)
```

La fonction *ldap_get_entries()* admet en paramètre l'identifiant du serveur LDAP retourné par la fonction *ldap_connect()* ainsi que l'identifiant du résultat retourné par la fonction *ldap_search()* et retourne un tableau multidimensionnel contenant le résultat de la recherche, c'est-à-dire le DN et le nom de l'entrée ainsi que la ou les valeurs de chacune d'entre-elles.

La structure du tableau retourné (on supposera qu'il se trouve dans la variable $entrees) est la suivante :

- $entrees["count"] : nombre d'entrées dans le résultat
- $entrees[0] : détail de la première entrée

- $entrees[i]["dn"] : DN de la i^{ème} entrée
- $entrees[i]["count"] : nombre d'attributs de la i^{ème} entrée
- $entrees[i][j] : valeur du j^{ème} attribut de la i^{ème} entrée
- $entrees[i]["attribut"] : valeur de l'attribut nommé "attribut" de la i^{ème} entrée (pour un attribut multivalué)
- $entrees[i]["attribut"]["count"] : Nombre de valeurs du j^{ème} attribut de la i^{ème} entrée (pour un attribut multivalué)
- $entrees[i]["attribut"][j] : J^{ème} valeur de l'attribut nommé "attribut" de la i^{ème} entrée (pour un attribut multivalué)

Les noms des attributs dans le tableau associatif sont en minuscules, ainsi l'attribut *givenName* devra être écrit *givenname* (par exemple *$entrees[i]["givenname"]*)

Voici un exemple d'utilisation :

```
<?php
$ds=ldap_connect($server);  // On suppose que le serveur LDAP est sur cet
hote
if ($ds) {
        $r=ldap_bind($ds,$rootdn,$rootpw);
        $dn = "dc=univ-lehavre,dc=fr";
        $filtre="(|(sn=$person*)(cn=$person*))";
        $restriction = array( "cn", "sn", "mail");
        $sr = ldap_search($ds, $dn, $filter);
        echo "Le nombre d'entrées retourné est de ".ldap_count_entries($ds,
$sr)."<br>";
           echo  "Récupération des entrées ...";
           $info = ldap_get_entries($ds, $sr);
           echo  "Affichage des données des ".$info["count"]. " entrées
trouvées :";
           for ($i=0; $i<$info["count"]; $i++)
                   {
                   echo  "<p align="justify">";
                   echo  "Le dn (Distinguished Name) est: ". $info[$i]["dn"]
."<br>";
                   echo  "Nom (sn) : ". $info[$i]["sn"][0] . "<br>";
                   echo  "Prénom (cn) : ". $info[$i]["cn"][0] . "<br>";
                   for($j=0;$j<$info[$i]["mail"]["count"];$j++) {
                        echo  "Email numéro $j: ". $info[$i][ "mail"]
[$j] ."<br>";
                }
                   }
           echo  "<p> ... Fermeture de la connexion";
        ldap_close($ds);
} else {
        echo "Connexion au serveur LDAP impossible";
}
```

```
?>
```