Annuaire Open LDAP sous Unix/Linux

Mise en oeuvre et administration

Version 26,1

Auteur : F. Micaux

formation@actilis.net

Table des matières

Notes personnelles

Notes personnelles

1- Concepts d'annuaire et notions de base

Notes personnelles

1.1- Service d'annuaire

1.1.1- Qu'est-ce qu'un annuaire ?

C'est un conteneur d'informations :
Elles sont organisées de manière arborescente
Chaque feuille a un type composé par des classes

Exemples d'annuaires courants : Des annuaires dits « offline »
Annuaire téléphonique
Carnet d'adresses
Catalogue

1.1.2- À quoi sert un annuaire « en ligne » ?

Chercher et trouver rapidement des informations,
Gérer des informations (carnet d'adresses, authentification, profils),

Les informations constituent une base de données, **plutôt optimisée pour la consultation**.

1.1.3- Un annuaire ne doit pas...

Remplacer un système de fichiers, un serveur de fichiers, une base de données relationnelle
Subir de fréquentes mises à jour (performances inadaptées en mise à jour)
Contenir des données volumineuses

Notes personnelles

1.1.4- Différents domaines d'application

Applications système (authentification, auto-montage, ...) :

L'annuaire peut être lié aux services réseaux d'entreprise

authentification et contrôle d'accès,

la localisation de serveurs de fichiers ou d'imprimantes

Il est alors étroitement lié au système d'exploitation

La plupart des systèmes "supportent / s'appuient sur" LDAP

Applications Intranet/Extranet/Internet :

Sert aux applications s'adressant aux utilisateurs finaux :

Gestion des profils utilisateurs,

Annuaire téléphonique ou annuaire de messagerie électronique,

Contrôle d'Accès à des pages web, SSO

Peut être la base d'information entre un fournisseur et ses clients,

Gestion d'abonnés, d'hébergements web ou messagerie, ...

Bases de données :

Peut fédérer un ensemble de bases de données gérées dans des SGBD séparés

Peut épauler un SGBD pour prendre en charge les aspects consultation

Notes personnelles

1.2- Un peu d'histoire

1.2.1- Bases de données historiques

Les bases des comptes utilisateurs des systèmes :

- Les fichiers plats d'Unix : **/etc/passwd**, et **/etc/group**
- La base des profils sur IBM MVS
- Les comptes Windows

Les bases de noms d'hôtes

- Sous Unix : **/etc/hosts**
- Internet Domain Name System (**DNS**) à partir de 1984
- Les bases de contacts des noms de domaines Internet : **Whois**

1.2.2- L'avant LDAP

Sous Unix,

- **NIS** (Yellow Pages), centralise des cartes (constituées à partir de fichiers plats)
- Bases de données pouvant être servies par NIS : /etc/hosts, /etc/passwd, /etc/group,

Sous Windows,

- notion de serveur d'authentification,
- notion de contrôleur de domaine,
- notion de serveur **WINS**

Notes personnelles

1.2.3- Le standard X500

Standard conçu pour l'interconnexion des annuaires téléphoniques des opérateurs télécom.

Normalisé et fédérateur, il a été conçu pour devenir **LE** standard d'annuaire global distribué

Conçu pour répondre à tout type de besoin d'annuaire…
grâce à un **modèle de données objet** extensible.

1.2.4- Particularités de X500

Principaux atouts :
distribué, ouvert,
dispose de fonctions de recherche avancées

Principaux défauts :
basé sur les protocoles ISO, contraires à la culture internet,
les implémentations sont très lourdes, et souvent buggées

X500 n'a pas atteint les objectifs escomptés

Notes personnelles

1.2.5- L'histoire de LDAP

Elle débute en 1993.

LDAP signifie : **Lightweight Directory Access Protocol**

Il est né de l'adaptation et du dégraissage de X.500 DAP au protocole TCP/IP.

Deux groupes de travail aboutissent à deux produits fonctionnant comme frontal X500 :
- DAS : Directory Assistance Service : RFC 1202
- DIXIE : Directory Interface to X500 Implemented Efficiently : RFC 1249

Ces deux produits convergent vers le standard IETF LDAP
- LDAPv1 : RFC 1487
- LDAPv2 : RFC 1777
- LDAPv3 : RFC 2251

LDAP, qui garde beaucoup de ressemblances avec X500, est orienté vers deux grands axes :
- simplification
- performances

Notes personnelles

1.3- Notions de conception LDAP

1.3.1- Réflexion préalable

Quelle est la nature des données hébergées ?
Comment récupérer les données ?
Quelle sera l'utilisation de ces données ?
Comment gère-t-on les données ?

1.3.2- Phases de conception

Quels sont les besoins ? Quelles applications actuelles ou futures accèderont aux données

Quelles sont les données nécessaires ? Inventaire et caractérisation des données
Format, taille, volume, contrôle d'accès, nature (dynamiques/statiques), partageables...

Modèle de nommage : comment les données sont organisées, nommées, accèdées

Topologie du service : performances, fiabilité, facilité de gestion...

On tiendra compte des applications et de leur nombre d'utilisateurs
des capacités du logiciel serveur, la topologie du réseau

Pour déterminer :
si la base sera centralisée ou répartie
le nombre de serveurs et leur emplacement sur le réseau

Notes personnelles

1.4- Les concepts de LDAP

1.4.1- Les modèles LDAP

Un **modèle de nommage** :
comment l'information est organisée et référencée,

Un **modèle fonctionnel** :
comment consulter et mettre à jour l'information,

Un **modèle d'information** :
les types d'informations (classes, attributs) stockés dans l'annuaire,

Un **modèle de sécurité** :
comment les accès aux données sont protégés,

Un **modèle de duplication** :
comment les données sont réparties / répliquées entre les serveurs,

1.4.2- Le format LDIF

LDIF : un format de données utilisé pour les sauvegardes / restaurations, l'import, la présentation...
LDAP **D**ata **I**nterchange **F**ormat.

Notes personnelles

1.4.3- Le protocole LDAP

LDAP est un **protocole** qui définit :

Les commandes de **communication entre clients et serveurs**
- **Connexion** & **déconnexion**
- **Recherche**, **consultation** et **comparaison**
- Mise à jour : **création**, **modification, renommage** et **suppression** de données

Les commandes de **communication entre serveurs**
- Échange de contenu et synchronisation (« replication service »)
- Création de lien entre serveurs pour relier les annuaires (« referral service »)

Le format de **transport des données**
- Le format ASCII n'est pas utilisé, à l'instar de HTTP, SMTP, ...
- Le format BER (Basic Encoding Rules) est utilisé par LDAP
  - Dans une forme allègée : LBER (Lightweight BER)

Les mécanismes de **sécurité**
- Méthodes de chiffrement et d'authentification (SASL…)
- Mécanismes et règles d'accès aux données (access to…)

Notes personnelles

1.4.4- Opérations LDAP

Le protocole définit aussi les **opérations de base**

Connexion au service :
- **bind**,
- **unbind**,
- **abandon**

Interrogation :
- **search**,
- **compare**

Mise à jour :
- **add**,
- **modify**,
- **rename**,
- **delete**

1.4.5- Les ports utilisés par LDAP

LDAP fonctionne sur le port **389/tcp** par défaut.

Le port **636/tcp** peut aussi être utilisé dans le cadre de LDAP**S**.

Notes personnelles

1.5- Les clients LDAP

Les commandes clientes : paquetage **openldap-clients** (CentOS), ou **ldap-utils** (Debian)

```
/usr/bin/ldapadd
/usr/bin/ldapcompare
/usr/bin/ldapdelete
/usr/bin/ldapexop
/usr/bin/ldapmodify
/usr/bin/ldapmodrdn
/usr/bin/ldappasswd
/usr/bin/ldapsearch
/usr/bin/ldapurl
/usr/bin/ldapwhoami
```

Authentification système : **sssd-ldap**[1],

Le paquetage **libuser** (gestion des comptes système LDAP),

Applications Web : Ldap Account Manager,

Les services : Samba, Apache, Postfix, Autofs, sudo…,

Les logiciels de messagerie et de contacts,

Les langages de programmation : C, Perl, PHP, ...et tout langage pouvant utiliser un SDK LDAP.

1 A remplacé depuis quelques années l'approche par les paquetages **nss-pam-ldapd** (RH/CentOS), ou **libnss-ldapd** (Debian)

Notes personnelles

2- Installation du serveur OpenLDAP

Notes personnelles

2.1- Paquetages OpenLDAP

2.1.1- Les composants d'OpenLDAP

OpenLDAP est une suite d'outils et de bibliothèques réparties en plusieurs paquetages

openldap / libldap : Bibliothèques nécessaires au serveur OpenLDAP et aux applications clientes
bibliothèques dynamiques, fichier de configuration client **ldap.conf**.

Openldap-clients / ldap-utils : commandes pour consulter et modifier l'annuaire LDAP.

Openldap-servers / slapd : Serveurs et utilitaires nécessaires au serveur LDAP.

des fichiers schémas standards, stockés dans le sous-répertoire **schema**

la configuration du serveur **slapd** :

historiquement fichier **slapd.conf(5)**
depuis la version 2.4 (Olc[2]) ⇒ dossier **slapd.d, …** man **slapd-config(5)**

le "script" (unité systemd) de démarrage / arrêt du service : **"slapd"**

des pages de manuel (dont un grand nombre dans la section 5 !)…

un ensemble d'exécutables de maintenance du serveur : commandes "slap*"

2 **O**n-**L**ine **C**onfiguration : repose sur un annuaire accessible localement uniquement par une branche "cn=config".

Notes personnelles

2.2- Versions de OpenLDAP

2.2.1- Versions / Roadmap

⇒ https://www.openldap.org/software/roadmap.html

2.2.2- Disponibilité du serveur

RedHat Entreprise (≥ 7.4) a déprecié[3] openldap-servers au profit de 389-DS, mais...

Disponibilités à fin 2024 :

Distribution	Repository	Nom	Version
Debian 12	Main	Slapd	2.5.13
Debian 11	Main	slapd	2.4.57
Ubuntu 22.04 (LTS)	Main	slapd	2.5.11
CentOS 7 (EOL 2022-06)	Base / updates	openldap-servers	2.4.44
RHEL 8 / Alma 8	Non disponible		
CentOS-Stream 8	Powertools	openldap-servers	2.4.46
CentOS-Stream 9 / Alma 9	EPEL	openldap-servers	2.6.2

3 https://access.redhat.com/solutions/2440481

Notes personnelles

2.3- Installation de OpenLDAP

2.3.1- Debian, Ubuntu et assimilées

Vérifier le nom de la machine

Le package Debian crée un premier annuaire qui en dépend
Spécifier le hostname (et le nom de domaine) : **hostnamectl set-hostname** <host.domain>
Vérifier **/etc/hostname** et/ou valider avec la commande "**hostname**",
Valider que ce nom est bien connu au niveau résolution de noms (DNS ou **/etc/hosts**).

Installation

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

L'installation prévoit la **saisie d'un mot de passe** pour l'administrateur du premier annuaire.

Notes personnelles

2.3.2- Dans le monde CentOS / Stream / Rocky / ALma

CentOS 7

```
# yum -y install openldap-clients openldap-clients openldap-servers
```

V8

```
# dnf config-manager --set-enabled powertools
# dnf -y install openldap-clients openldap-servers
```

V9

```
# dnf -y install epel-release
# dnf -y install openldap-clients openldap-servers
```


Notes personnelles

2.4- Démarrage du service

Démarrage

```
# systemctl enable --now slapd
```

Vérification du démarrage

```
# pgrep -a slapd
2363 /usr/sbin/slapd -u ldap -h ldap:/// ldaps:/// ldapi:///
```

Sockets

ldap:/// tcp/389, utilisé par les clients non TLS
ldaps:/// tcp/636, utilisé par les clients TLS
ldapi:/// socket unix (/var/run/ldapi), utilisée pour l'administration du serveur.

Le premier annuaire

Sur Debian, à l'installation du package, et un premier annuaire est créé.

Sa base est dérivée du nom de domaine de la machine.
L'administrateur est "cn=admin,dc=….." et le mot de passe a été choisi pendant l'installation.

Dans le monde CentOS, un premier annuaire sera créé au premier démarrage...

Sa base est par défaut "dc=my-domain,dc=com",
l'administrateur est "cn=Manager, dc=my-domain,dc=com", il n'a pas de mot de passe.

Notes personnelles

2.5- Éléments de configuration

Configuration
Depuis la version 2.4 la configuration est stockée dans uen branche "cn=config". (man **slapd-config**)

Emplacement des fichiers de configuration
Sous Debian : **/etc/ldap/slapd.d/**…
Sous CentOS/RHEL//Rocky/Alma : **/etc/openldap/slapd.d/**…

On ne modifie surtout pas manuellement ces fichiers par "vi" (ni par "nano", ni par "micro"...),
⇒ mais grâce à des commandes "**ldapmodify/ldapadd**"
⇒ ou au travers de l'utilitaire "**ldapvi**".

Répertoire des données et format
Les données de l'annuaire par défaut se trouvent dans **/var/lib/ldap**.

Il existe plusieurs formats de stockage : **HDB** et **MDB**
⇒ Debian utilise le format MDB (celui recommandé depuis quelques années),
⇒ RedHat/CentOS utilise par défaut HDB (CentOS <=7) et MDB (CentOS Stream >=8).

Notes personnelles

3- Concepts fondamentaux LDAP

Notes personnelles

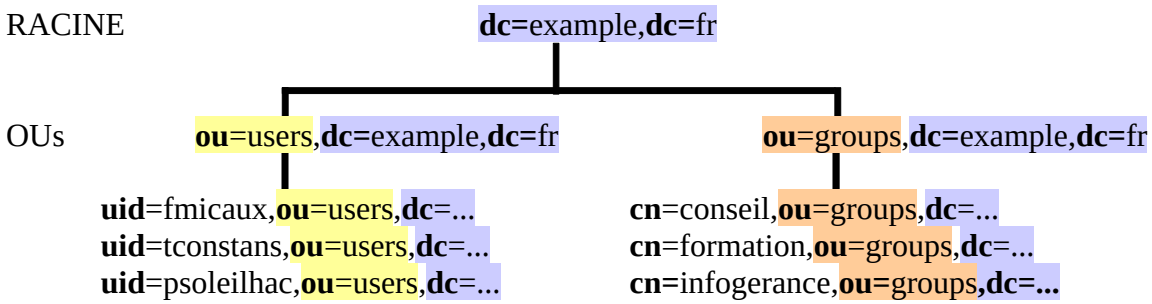
3.1- Modèle de nommage

Le **modèle de nommage** définit
comment sont **organisés** les éléments de l'annuaire
comment ils sont **référencés, identifiés**

3.1.1- Représentation hiérarchique

Cette organisation s'appelle **Directory Information Tree** (DIT), toutes ses données ont la même racine.

Représentation « visuelle » :
L'identifiant de chaque objet est son nom distinctif, le **DN** (Distinguish Name).



3.1.2- Filiation

Un élément marque son appartenance à sa hiérarchie en reprenant son nom (**DN** du père)...
...qu'il complète par le sien (**RDN** de l'entrée + **DN** du père = **DN** de l'entrée).

Notes personnelles

3.1.3- Éléments et identifiants

Chaque élément est appelé une entrée (**entry**) et peut être :

un **nœud** (**node**) : branchement (un contenant pour d'autres entrées),
un nœud peut disposer de propriétés (attributs), car c'est aussi un objet.

une **feuille** (**leaf**) : un élément terminal
une feuille est un objet qui n'est le père d'aucun autre.

3.1.3.1- La racine

C'est l'élément initial de l'arbre, aussi appelé le **root DN** (DN Racine) ou "suffixe".

La racine est souvent composée de plusieurs attributs **DC** (**Domain Component**)

Exemple : **dc=**example,**dc=**fr

3.1.3.2- Unités d'Organisation : OU

Des objets de type **OU** (**Organizational Unit**) sont souvent présents entre la racine et les feuilles.

Souvent, on a même une arborescence composée de plusieurs OU
L'OU est un peu le pendant des répertoires dans l'arborescence Unix.

Notes personnelles

3.1.3.3- DN & RDN

Chaque entrée possède un **DN** (**Distinguish Name**) : c'est son **nom complet**

Il est **unique dans l'annuaire**,

Il permet de positionner / désigner l'élément dans l'annuaire
On ne peut pas insérer un objet dans l'annuaire sans lui affecter un DN.

Un utilisateur qui se connecte à l'annuaire utilise son DN comme identifiant.

Chaque élément possède un **RDN** (**Relative Distinguish Name**) :

C'est la partie de son DN relative au DN de son supérieur
Il n'est pas forcément unique, et ne permet pas d'identifier un élément de manière absolue

Règle de nommage (introduite par la RFC 2253)

Pas d'espace de part et d'autre des virgules séparant les types d'entité.

DN : cn=fmicaux,ou=users,dc=example,dc=fr
mais pas
DN : cn=fmicaux , ou=users , dc=example , dc=fr

Notes personnelles

3.2- Modèle fonctionnel

Il définit les **moyens d'accès** aux données, et les **opérations** qu'elles peuvent subir :

authentification et **contrôle**, **opérations étendues** (ldapv3)

interrogation, **comparaison**, **mise à jour**,

3.2.1- Opérations d'authentification

Il existe 3 opérations : **bind**, **unbind**, **abandon**

bind : connexion à l'annuaire

unbind : déconnexion

abandon : le client annonce au serveur qu'il abandonne la requête en cours

le serveur abandonne alors le processus en train de l'exécuter

Notes personnelles

3.2.2- Opérations de mise à jour

Il existe 4 opérations : **add**, **modify**, **rename**, **delete**

Elles doivent respecter des droits d'accès et des pré requis :

add, **rename** : l'entrée ne doit pas déjà exister, et doit posséder un parent existant

add, **modify** : les attributs doivent être conformes au schéma

delete : l'entrée ne doit pas avoir d'enfant

3.2.3- Opérations de recherche et de comparaison

L'opération de **recherche** renvoie l'entrée si elle trouve,

L'opération de comparaison **vérifie** si un attribut d'une entrée contient ou non une valeur.
Elle renvoie **vrai** ou **faux**

Dans le cas où l'attribut n'existe pas
la **recherche** ne renvoie rien,
la **comparaison** retourne une erreur.

L'opération de comparaison est un héritage de X500.

Notes personnelles

3.2.4- Opérations de recherche et comparaison : portée

Les opérations sont réalisées **à partir d'une base** et **pour une certaine portée**

Base : DN à partir duquel l'opération doit agir.

Il y a toujours une base dans les opérations de recherche.

Scope (portée) : nombre de niveaux sur lequel l'opération doit agir.

base : l'action est réalisée uniquement sur la base.

sub : récursion à partir de la base, sur la totalité de l'annuaire,

one : fils directs de la base,

Notes personnelles

3.2.5- Opérations de recherche et comparaison : filtres

Les opérations peuvent utiliser des **filtres** (opérateurs), pour effectuer des tests de correspondance :

4 tests basiques peuvent être combinés :

=	(égal), * : joker.
<=	(plus petit ou égal),
>=	(plus grand ou égal),
~=	(approximation)

Pas d'opérateur < ou > (stricte), mais possibilités de combinaisons avec :

( )	les parenthèses pour délimiter des expressions,
&	(et, intersection),
|	(ou, union),
!	(non)

Exemple : pour effectuer un test d'infériorité stricte (A < B) :

(&(A<=B)(!(A=B))) : A est inférieur ou égal à B, ET A n'est pas égal à B.

Notes personnelles

3.2.6- Les URLs LDAP

C'est une méthode définie par la RFC 2255 pour formuler des interrogations de l'annuaire.

Syntaxe :

```
ldap[s]://serveur[:port][/[base[?[attributs][?[portée][?[filtre][?extensions]]]]]]
```

Ces URLs ne sont pas prises en compte par les outils standards (ldapsearch, firefox, …).

Elles sont utilisables dans certains logiciels pour formuler des requêtes.

Par exemple : dans Apache, (mod_authnz_ldap : **AuthLdapUrl**),

Exemple : Recherche des champs **uid** de l'arbre

```
AuthLDAPUrl ldap://ldap.formation.actilis.fr:389/dc=example,dc=fr?uid?sub
```


Notes personnelles

3.3- Modèle d'information

Le modèle d'information repose sur des schémas.

3.3.1- Les schémas

C'est l'**ensemble exhaustif** des définitions relatives aux objets que l'annuaire sait gérer,

Les **schémas** définissent les **Classes d'objets** et les **Attributs autorisés dans l'annuaire**.

Les **classes d'objets** définissent la nature des objets qui peuvent exister dans l'annuaire

Les **attributs** définissent les caractéristiques de chaque classe d'objet,

Le schéma décrit :

les **classes d'objets**,
les **types des attributs**,
la **syntaxe des attributs**

Un fichier de schéma sera chargé par le serveur LDAP au démarrage

On peut bien sûr en charger plusieurs.

Notes personnelles

3.3.2- Classes d'objet

Les **objectClass** servent à cataloguer les entrées, définir leur type :

Un **objectClass** définit un **groupe d'attributs** obligatoires ou autorisés dans une entrée,

Une entrée doit contenir au moins une déclaration **objectClass**,
elle ne peut porter que les attributs appartenant à ces différentes classes d'objets,

Exemple : l'objectClass "**posixAccount**"

```
objectclass ( 1.3.6.1.1.1.2.0 NAME 'posixAccount'
        DESC 'Abstraction of an account with POSIX attributes'
        SUP top AUXILIARY
        MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
        MAY ( userPassword $ loginShell $ gecos $ description ) )
```

MUST présente des attributs **obligatoires**.
MAY présente des attributs **optionnels**.

Plusieurs fichiers de schéma sont fournis dans le sous-répertoire "**schema**" (/etc/openldap/schema).

Il faut forcément charger ceux décrivant les objets qu'on souhaite utiliser.

Notes personnelles

3.3.3- Les attributs

Chaque attribut possède une description (**DESC**...)…

...et est défini par :

un type d'égalité à mettre en œuvre lors d'une recherche qui le concerne
EQUALITY...,

ainsi que la syntaxe du type de données qu'il contient
SYNTAX ...

```
attributetype ( 1.3.6.1.1.1.1.0 NAME 'uidNumber'
        DESC 'An integer uniquely identifying a user in an administrative domain'
        EQUALITY integerMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )
```

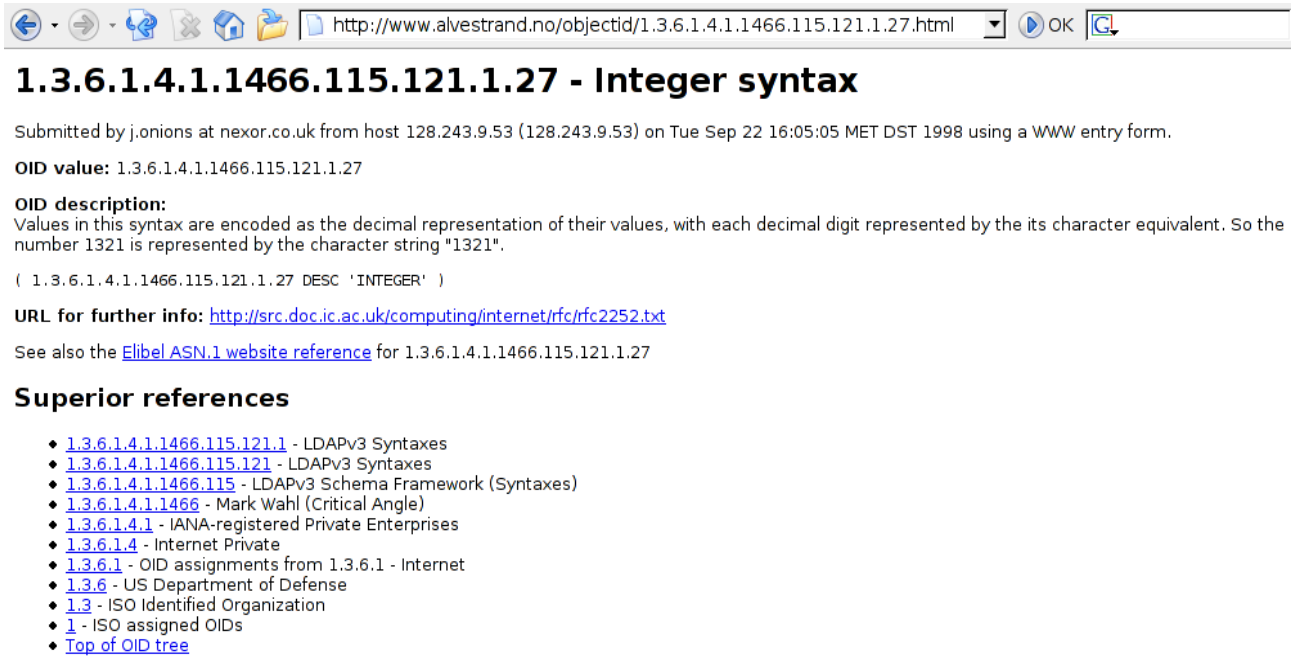
3.3.4- OID : Object Identifier

Chaque attribut possède un **OID**, et est typé. Chaque type a aussi un **OID.**

Cela permet de rendre unique l'attribut spécifié.
Les OIDs sont déposés auprès de l'IANA (www.iana.org) et sont donc officiels.
Un site pour les consulter : http://www.alvestrand.no/objectid/1.3.6.1.1.1.html

Notes personnelles

La référence **1.3.6.1.4.1.1466.115.121.1.27** est un OID, c'est la définition du type « integer »



Notes personnelles

3.3.5- Attributs des objectClass

Tout **objectClass** possède aussi un **OID**... ainsi que plusieurs autres caractéristiques :

```
objectclass ( 1.3.6.1.1.1.2.0 NAME 'posixAccount'
        DESC 'Abstraction of an account with POSIX attributes'
        SUP top
        AUXILIARY
        MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
        MAY ( userPassword $ loginShell $ gecos $ description )    )
```

NUMERIC OID (mandatory)	Unique object identifier (OID)
NAME	Nom de l'objectClass
DESC	Simple description
OBSOLETE	"true" si obsolete; "false" ou absent sinon
SUP	Nom de l'objectClass dont dérive l'objetClass (top sinon)
ABSTRACT	"true" si abstrait; "false" ou absent sinon
STRUCTURAL	"true" si structural; "false" ou absent sinon
AUXILIARY	"true" si auxiliaire; "false" ou absent sinon
MUST	Liste des attributs qui doivent être présents dans une entrée
MAY	Liste des attributs qui peuvent être présents dans une entrée

Notes personnelles

3.3.6- Abstract, Structural, Auxiliary ?

3.3.6.1- ABSTRACT

Le type ***ABSTRACT*** permet juste de définir une classe dont doivent dériver d'autres classes.

Une classe de type ***ABSTRACT*** ne peut avoir aucune instance d'objet dans l'annuaire.

3.3.6.2- STRUCTURAL

Le type ***STRUCTURAL*** permet de définir une classe qui dérive d'une autre,

Pour exister dans l'annuaire, un objet doit être appuyé sur une classe de type STRUCTURAL

Les classes afférentes aux groupes et utilisateurs sont en principe « structurelles ».

La classe racine est la classe **top**, (de type ***ABSTRACT***).

3.3.6.3- AUXILIARY

Le type ***AUXILIARY*** permet de s'affranchir de ce mécanisme d'héritage,

et d'attribuer des données complémentaires à un objet.

Notes personnelles

3.3.7- Les entrées de l'annuaire

L'entrée (**Entry**) est l'élément de base de l'annuaire, elle contient toujours :

Un **nom distinctif** (dn)

Les **classes** dont l'entrée dépend (objectClass)

Les **attributs et leurs valeurs** (cn, uid, uidNumber…)

Les attributs utilisables ne sont pas libres mais dépendent de la classe de l'objet.

Certains sont **obligatoires**, d'autres sont **permis**,

Dans le format LDIF, un attribut est séparé de sa valeur par le caractère "**:**"

```
dn: uid=fmicaux,ou=users,dc=example,dc=fr
objectClass: account
objectClass: posixAccount
cn: fmicaux
uid: fmicaux
uidNumber: 1000
…
```


Notes personnelles

3.3.8- Le format LDIF

LDIF (**L**DAP **D**ata **I**nterchange **F**ormat (RFC 2849)).

C'est le standard de représentation des entrées sous forme texte.

Il est utilisé pour afficher ou modifier des données de la base, selon deux modes :
- imports/exports (et donc sauvegarde / restauration), duplications
- réaliser des modifications sur des entrées

On parle de fichiers LDIF, ce sont des **fichiers texte**
- Le format utilisé dans un fichier LDIF est l'ASCII
- Toute donnée (valeur d'attribut ou DN) qui n'est pas ASCII est codée en base 64.

Une entrée LDIF ressemble un peu à ceci :

```
# [Commentaire si nécessaire]
dn: <distinguished name>
<attrtype>: <attrvalue>
<attrtype>: <attrvalue>
<attrtype>: <attrvalue>
--- LIGNE VIDE = FIN D'ENTRÉE ---
```

Toutes les paires **<attrtype>** et **<attrvalue>** doivent être définies par un fichier de schéma.

Notes personnelles

3.4- Modèle de sécurité

3.4.1- Plusieurs niveaux de sécurité

Authentification pour se connecter au service, et bénéficier de droits sur les objets.

Modèle de **contrôle d'accès aux données**

Chiffrement des transactions client-serveur ou serveur-serveur,

3.4.2- Authentification

LDAP est un protocole avec connexion, dont l'ouverture de session (**bind**) nécessite :
une **identification** (le DN d'un compte connu de l'annuaire)
éventuellement (v3) **un mot de passe** (celui associé au DN spécifié)

Plusieurs types d'authentification sont supportés
Anonymous Authentication (v2 & v3) : accès sans restriction ni authentification,
Root DN Authentication (v2 & v3) : accès administrateur (tous les droits sont accordés),
Clear Password (v2 & v3) : un DN + un password qui transitent en clair,
Kerberos v4 (v2),
TLS ou **LDAPS** (Password + SSL) : session chiffrée, le password ne transite plus en clair,
Certificats SSL : échange de certificats (clé publique / clé privée),
Simple **A**uthentication and **S**ecurity **L**ayer (v3) : mécanisme externe d'authentification,

Notes personnelles

3.4.3- Contrôle d'accès

Le serveur attribue des droits d'accès sur les données à un utilisateur authentifié
- Ils concernent les opérations de recherche, comparaison et mise à jour
- Ils sont définis à l'aide de listes de contrôle d'accès (ACL)
- Chaque ACL porte sur une branche particulière de l'annuaire

Le contrôle d'accès n'est pas normalisé par IETF,
- donc pas forcément compatible entre toutes les implémentations
  - Netscape Directory les code sous forme d'un attribut dans les objets (**A**ccess **C**ontrol **I**tem)
  - OpenLDAP les code sous forme de directives de type « access to »

3.4.4- Chiffrement et certificats

LDAPv3 supporte le chiffrement des transactions
- client-serveur
- serveur-serveur

SSL (dans le cas de ldaps) ou son successeur **TLS** (**T**ransport **L**ayer **S**ecurity) sont utilisés.

SSL ou TLS servent également dans l'authentification par certificats
- Permet au client de prouver son identité
- Permet au serveur de faire de même en retour

Notes personnelles

3.5- Modèle de réplication

3.5.1- Réplication : dans quel but ?

Panne sur un serveur, coupure réseau, surcharge du service… l'objectif est d'améliorer la disponibilité du service et ses temps de réponse.

La **réplication (ou duplication)** permet entre autre :

- d'améliorer les performances en plaçant les serveurs près des clients (service de proximité)
- de répartir la charge entre plusieurs serveurs (Load Balancing)
- de gérer les entrées localement et de les répartir ensuite (système distribué)

La duplication LDAP est définie par la RFC 3384.

- Le format d'échange est **LDIF**.

3.5.2- Deux types de réplication

Le mode « **maître-esclave** »

- Unidirectionnel, le maître envoie les modifications à l'esclave
- Les écritures ne sont donc autorisées que sur le maître
- L'esclave est en « lecture seule »

Le mode « **maître-maître** »

- Bi-directionnel, chacun étant maître de l'autre annuaire
- On peut intervenir en écriture sur chacun des annuaires

Notes personnelles

4- Manipuler le contenu d'un annuaire

Notes personnelles

4.1- Commandes clientes / outils d'administration

4.1.1- Les commandes en "slap*" : outils d'administration

Les commandes en "**slap***" accèdent directement aux bases de données

Sauvegarder / restaurer : **slapcat**, **slapadd**
Manipuler les index : **slapindex**

Ces commandes s'utilisent en principe annuaire éteint (slapadd)
ou accessible en lecture seule (slapcat, slapindex).

4.1.2- Les commandes en "ldap*" : commandes clientes

Toutes les commandes clientes en "**ldap***" utilisent le **protocole LDAP**.
Elles sont fournies par le package **openldap-clients** (ldap-utils).

Elles peuvent donc être utilisées depuis toute machine.
Elles impliquent que le serveur d'annuaire contacté soit démarré.

Requêter dans le contenu :
ldapsearch, **ldapcompare**

Manipuler le contenu :
ldapadd, **ldapdelete**, **ldapmodify**, **ldapmodrdn**

Notes personnelles

4.2- Panorama des commandes ldap*

Utilitaires de recherche dans l'annuaire :
ldapsearch : rechercher des entrées dans l'annuaire
Peut utiliser les filtres de recherche (un par ligne) stockés dans le fichier précisé par « **-f fic** »,
Peut utiliser les filtres saisis sur son entrée standard si le nom du fichier est « - » : **-f -**

Utilitaires de mise à jour de l'annuaire :
ldapmodify : modifier des entrées de l'annuaire.
L'entrée peut être spécifiée par un fichier LDIF ou sur l'entrée standard de la commande

ldapadd : ajouter des entrées dans l'annuaire.
L'entrée peut être spécifiée par un fichier LDIF ou sur l'entrée standard de la commande
La commande **ldapadd** est en fait un lien vers **ldapmodify**. (équivalent à **ldapmodify -a**)

ldapdelete : supprimer des entrées de l'annuaire
L'entrée peut être spécifiée par un fichier LDIF ou sur l'entrée standard de la commande

Utilitaires divers :
ldapwhoami : affiche l'identité (avec laquelle le binding a eu lieu)
ldappasswd : modifie le mot de passe d'une entrée de l'annuaire
ldapmodrdn : renomme une entrée
ldapcompare : compare la valeur d'un attribut d'une entrée à une valeur arbitraire

Notes personnelles

4.3- Tester l'accès à un annuaire : ldapwhoami

Quelques options de **ldapwhoami** :

-H : désigne le serveur grâce à une URL indiquant protocol, host, et port
- **ldapi:///** socket locale /var/run/ldapi (/var/run/slapd/ldapi sous Debian)
- **ldap://x.y.z.t/** socket réseau (tcp/389)
- **ldaps://x.y.z.t/** socket réseau (tcp/636) et chiffrement TLS

-Y EXTERNAL : mécanisme d'authentification spécifique basé sur l'OS
Ne fonctionne **que localement** (-H ldapi:///) potentiellement pour tout utilisateur Unix.

-D « DN » : pour spécifier le DN de connexion (le login...)
-w mot-de-passe : pour donner le mot de passe sur la ligne de commande
-**W** pour demander une saisie interactive, **-y** pour le prendre dans un fichier.

```
# ldapwhoami -Q -Y EXTERNAL -H ldapi:///
dn:gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth

# ldapwhoami  -H ldap:/// -D "cn=manager,dc=example,dc=fr" -w 123Soleil
dn:cn=manager,dc=example,dc=fr

# ldapwhoami  -H ldap://127.0.0.1:389/ -D "cn=manager,dc=example,dc=fr" -w 123Soleil
dn:cn=manager,dc=example,dc=fr

# ldapwhoami -D "" anonymous
```


Notes personnelles

4.4- Rechercher des entrées : ldapsearch

La commande **ldapsearch** permet de réaliser des recherches dans l'annuaire.

```
$ ldapsearch -x -h <serveur> -b <base> [-s portée] [filtre] [attributs ...]
```

On doit préciser : La **base** de la recherche (-b)

On peut préciser :

La **portée** de la recherche (-s avec **base**, **one**, **sub, children**), "sub" par défaut.

Le **filtre** de recherche (quotes recommandées à cause des *)

Les **attributs** à afficher (séparés par espace).
Tous ("*****") sont affichés par défaut
⇒ Afficher les attributs opérationnels : "+".
⇒ Tous ainsi que les attributs opérationnels : **"*"** +
⇒ Seulement les attributs d'une classe : **@classe**

Si l'annuaire n'est pas accessible publiquement, on doit s'authentifier :**-D** DN, **-W** / -w / **-Y**[4]
on utilise **-D ""** pour une "authentification" de type ***anonymous***

4 En principe, on réserve l'authentification -Y EXTERNAL à la modification de cn=config. On utilisera de préférence le RootDN de l'annuaire ou le DN d'une entrée existant dans l'annuaire pour l'administrer, requêter....

Notes personnelles

Exemples

Lister les entrées dont le champ gecos contient « lisat »

```
$ ldapsearch -x -h vm1 -b dc=example,dc=fr '(gecos=*lisat*)'
```

Lister uniquement les champs uid et uidNumber de toutes les entrées à partir de ou=users,...

```
$ ldapsearch -x -h vm1 -b ou=users,dc=example,dc=fr uid uidNumber
```

Lister uniquement les attributs de classe olcMdbConfig des entrées décrivant une base de données :

```
# ldapsearch -QLLL -Y EXTERNAL -H ldapi:/// -b cn=config "(objectClass=olcDatabaseConfig)"
@olcMdbConfig
```

Uniquement les attributs opérationnels pour "dc=example,dc=fr" :

```
# ldapsearch -LLL -D "" -b "dc=example,dc=fr" +
dn: dc=example,dc=fr
structuralObjectClass: organization
entryUUID: 9b09348c-e014-103c-8e35-0dccb15d3b04
creatorsName: cn=manager,dc=example,dc=fr
createTimestamp: 20221014140238Z
entryCSN: 20221017071039.779651Z#000000#000#000000
modifiersName: cn=manager,dc=example,dc=fr
modifyTimestamp: 20221017071039Z
entryDN: dc=example,dc=fr
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```


Notes personnelles

4.5- Ajouter des entrées : ldapadd

La commande **ldapadd** permet d'ajouter des entrées dans l'annuaire.

Les options d'authentification sont similaires à celles de **ldapwhoami**. En complément, on utilise "-f" :

-f fichier.ldif : c'est le fichier contenant les données à ajouter.

Le contenu peut aussi être fourni sur l'entrée standard (avec redirection <).

Fichier ou-test.ldif :

```
dn: ou=test,dc=example,dc=fr
objectClass: top
objectClass: organizationalUnit
ou: test
```

4.5.1- L'erreur "parent manquant"

Exemple avec un échec :

```
# ldapadd -D  cn=manager,dc=example,dc=fr -w passadmin   -f ou-test.ldif
adding new entry "ou=test,dc=example,dc=fr"
ldap_add: No such object (32)
```

L'ajout a échoué… **No such object** : On a tenté d'ajouter un objet dans un nœud qui n'existe pas.

l'annuaire est tout simplement vide et n'a pas d'entrée racine !

Notes personnelles

Création d'une racine de type "Organization" :

Le fichier "org-dom1.ldif" est créé avec le contenu suivant :

```
dn: o=dom1
objectClass: organization
o: dom1
```

La propriété "o" est la seule obligatoire pour la classe "organization".

4.5.2- L'erreur "le DN cité n'est pas autorisé dans cet annuaire"

Tentative d'ajout :

```
# ldapadd -D  cn=manager,dc=example,dc=fr -w passadmin   -f org-dom1.ldif
adding new entry "o=dom1"
ldap_add: Server is unwilling to perform (53)
       additional info: no global superior knowledge
```

Echec ! **No Global Superior knowledge**

Le DN est non valide (la base de l'annuaire est dc=example,dc=fr)
Il faudrait que le DN de l'entrée racine de cet annuaire soit "**dc=example,dc=fr**".

Notes personnelles

Modification du DN de l'enregistrement :

Le fichier "org-dom1.ldif" est modifié avec le contenu suivant :

```
dn: dc=example,dc=fr
objectClass: organization
o: dom1
```

4.5.3- L'erreur "un attribut n'est pas connu"

Tentative d'ajout :

```
# ldapadd -D  cn=manager,dc=example,dc=fr -w passadmin   -f org-dom1.ldif
adding new entry "dc=example,dc=fr"
ldap_add: Object class violation (65)
        additional info: attribute 'dc' not allowed
```

Echec ! **Attribute "dc" not allowed**

Cet attribut n'est pas prévu par l'objectClass "organization".

Il faut trouver une classe complémentaire qui permet de l'utiliser
Recherche dans les schémas... ⇒ **dcObject**.

Notes personnelles

4.5.4- Un ajout réussi

Ajout de la classe "dcObject" à l'enregistrement :
Le fichier "org-dom1.ldif" est modifié avec le contenu suivant :

```
dn: dc=example,dc=fr
objectClass: organization
objectClass: dcObject
o: dom1
```

Cette-fois-ci, c'est bon, la racine peut être créée

```
# ldapadd -D  cn=manager,dc=example,dc=fr -w passadmin   -f org-dom1.ldif
adding new entry "dc=example,dc=fr"
```

On peut maintenant tenter l'ajout de l'OU "test", puisque la racine (le parent) existe.

```
# ldapadd -D  cn=manager,dc=example,dc=fr -w passadmin   -f ou-test.ldif
adding new entry "ou=test,dc=example,dc=fr"
```


Notes personnelles

4.5.5- L'erreur "Déjà existant"

Si une entrée existe déjà : Erreur "**Already exists**"

```
# ldapadd -D  cn=manager,dc=example,dc=fr -w passadmin   -f ou-test.ldif
adding new entry "ou=test,dc=example,dc=fr"
ldap_add: Already exists (68)
```

Dans ce cas, ldapadd s'arrête à l'entrée générant l'erreur.

On peut forcer ldapadd à conteinuer en cas d'erreur avec l'option "-c" (continous)

Dans ce cas, il n'y a pas d'insertion pas en doublon,
mais il continue la lecture du fichier LDIF pour insérer les enregistrements suivants.

Notes personnelles

4.5.6- Exemple d'un DIT typique

Pour la gestion d'utilisateurs et de groupes, utilisateurs et groupes ont chacun leur OU...

Ce DIT pourrait convenir, à condition de paramétrer les applications en conséquence :

La **racine** de l'annuaire (dn: dc=example,dc=fr)

L'**OU users** (dn: ou=users,dc=example,dc=fr)

L'**OU groups** (dn: ou=groups,dc=example,dc=fr)

On peut utiliser un même fichier LDIF pour insérer plusieurs entrées à la suite.

Proposition de fichier « arbre-initial.ldif »

```
dn: dc=example,dc=fr
objectClass: dcObject
objectClass: organization
dc: example
o: Example
description: Exemple de description

dn: ou=users,dc=example,dc=fr
objectClass: top
objectClass: organizationalUnit
ou: users

dn: ou=groups,dc=example,dc=fr
objectClass: top
objectClass: organizationalUnit
ou: groups
```


Notes personnelles

Insertion des données (par ldapadd)

```
$ ldapadd -h vm1 -D "cn=ldapadmin,dc=example,dc=fr" -x -W -f arbre-initial.ldif
Enter LDAP Password:
adding new entry "dc=example,dc=fr"

adding new entry "ou=users,dc=example,dc=fr"

adding new entry "ou=groups,dc=example,dc=fr"
```

On peut ensuite passer à la création de "usera" :

```
$ ldapadd -h vm1 -D "cn=ldapadmin,dc=example,dc=fr" -x -W -f usera.ldif
Enter LDAP Password:
adding new entry "uid=usera,ou=users,dc=example,dc=fr"
```

Le fichier usera.ldif

```
dn: uid=usera,ou=users,dc=example,dc=fr
objectClass: account
objectClass: posixAccount
cn: usera
uid: usera
uidNumber: 10001
gidNumber: 9999
homeDirectory: /home/usera
userPassword: $1$HlJhmOER$Uzwdkka31qrYh/ohGheMH0=
loginShell: /bin/bash
gecos: Utilisateur A
description: usera
```


Notes personnelles

4.6- Supprimer des entrées : ldapdelete

La commande **ldapdelete** permet de supprimer des entrées de l'annuaire.

```
~$ ldapdelete -h vm1 -D "cn=ldapadmin,dc=example,dc=fr" -x -W uid=usera,ou=users,dc=example,dc=fr
Enter LDAP Password:
```

On spécifie **le DN de l'entrée à supprimer**.
Ou **une liste de DN à supprimer**,
ou rien, et les DN à supprimer seront lus sur l'entrée-standard.

Récursivité
L'option **-r** permet de supprimer une branche entière, **récursivement**, à partir du DN spécifié.

Pour supprimer tout l'annuaire
Supprimer tout l'annuaire en utilisant la récursivité depuis la racine est parfois très long.

La méthode plus rapide et plus brutale, est la suivante :
Arrêter le serveur
Supprimer tout le contenu du répertoire mentionné par "olcDbDirectory"
Redémarrer les serveur

Notes personnelles

4.7- Modifier le contenu de l'annuaire : ldapmodify

La commande **ldapmodify** permet de modifier des entrées de l'annuaire.

Elle réalise en fait des modifications de l'annuaire, et pas seulement des enregistrements.

Elle permet donc :

d'ajouter et de supprimer des enregistrements… (comme ldapadd et ldapdelete)

d'ajouter des attributs, d'en modifier, d'en supprimer...

Ajouter userb grâce à **ldapmodify** :

```
~$ ldapmodify -h vm1 -D "cn=ldapadmin,dc=example,dc=fr" -x -W -f userb.ldif
Enter LDAP Password:
```

Elle utilise un fichier LDIF qui doit spécifier

le **DN**,

l**e type d'opération** réalisée,

changetype: **add** : pour ajouter une entrée

changetype: **delete** : pour supprimer une entrée

changetype: **modify** : pour modifier une entrée déjà présente

La ligne du dessous mentionne alors le type de modification :

add : attribut

delete : attribut

replace : attribut

La ligne du dessous donne alors la nouvelle valeur de l'attribut.

Notes personnelles

Exemples :

Suppression d'un enregistrement :

```
dn: uid=userb,ou=users,dc=example,dc=fr
changetype: delete
```

Ajout de userb avec ldapmodify

```
dn: uid=userb,ou=users,dc=example,dc=fr
changetype: add
objectClass: account
objectClass: posixAccount
cn: userb
uid: userb
uidNumber: 10002
gidNumber: 9999
homeDirectory: /home/userb
userPassword: $1$HlJhmOER$Uzwdkka31qrYh/ohGheMH0=
loginShell: /bin/bash
gecos: Utilisateur B
description: userb
```

Ajout d'un attribut « description » :

```
dn: uid=userb,ou=users,dc=example,dc=fr
changetype: modify
add: description
description: Utilisateur B deuxieme description
```


Notes personnelles

Supprimer un l'attribut "description". S'il y en a plusieurs, on précise lequel

```
dn: uid=userb,ou=users,dc=example,dc=fr
changetype: modify
delete: description
description: userb
```

Modifions l'attribut description (= ajouter le nouveau + supprimer l'ancien, mais plus rapidement) :

```
dn: uid=userb,ou=users,dc=example,dc=fr
changetype: modify
replace: description
description: Utilisateur B
```

Si nous avions eu deux « descriptions », les deux seraient impactées par cette dernière opération.

On peut enchainer un « delete » et un « add » dans la même opération de « modify » :

Il suffit de séparer les deux actions par un « - »

```
dn: uid=userb,ou=users,dc=example,dc=fr
changetype: modify
delete: description
description: Utilisateur B
-
add: description
description: Utilisateur B nouvelle
```


Notes personnelles

4.8- Renommer une entrée avec ldapmodrdn

On spécifie le **DN de l'entrée** à modifier ainsi que le **nouveau RDN**

```
~$ ldapmodrdn -h vm1 -D "cn=ldapadmin,dc=example,dc=fr" -x -W 'uid=userb,ou=users,dc=example,dc=fr'
'uid=userc'
Enter LDAP Password:
```

L'ancien attribut « uid=userb » est toutefois conservé, comme l'attribut « cn=userb »...

```
~$ ldapsearch -x -h vm1 -b dc=example,dc=fr '(uid=*userc*)'
# extended LDIF
…
# userc, users, example.fr
dn: uid=userc,ou=users,dc=example,dc=fr
objectClass: account
objectClass: posixAccount
cn: userb
uid: userb
uid: userc
…
```

L'option « -s » permet de déplacer un objet : (userzzz migre de l'OU users vers la racine)

```
~$ ldapmodrdn -h vm1 -D "cn=ldapadmin,dc=example,dc=fr" -x -W -s dc=example,dc=fr
'uid=userzzz,ou=users,dc=example,dc=fr' 'uid=userzzz'
```


Notes personnelles

5- Configurer et Administrer le serveur OpenLDAP

Notes personnelles

5.1- Fichiers de configuration

5.1.1- Configuration du client

Le fichier **ldap.conf** est celui utilisé par tous les outils clients LDAP.

Il définit l'annuaire auquel on se connecte si on ne le précise pas dans les commandes.
Il est installé en standard car fourni par le paquetage openldap.
Il doit être world-readable, mais modifiable par « root » uniquement.

Une version minimale de celui-ci pourrait être la suivante

```
## See ldap.conf(5) for details

HOST 127.0.0.1
BASE dc=example,dc=fr
```

5.1.2- Configuration du serveur

Historiquement, on travaillait sur un fichier "**slapd.conf**" (voir en annexe).

Depuis la version 2.4, on utilise "**olc**[5]", qui permet une configuration dynamique
"**olc**" repose sur un répertoire "**slapd.d**".
On peut créer "**slapd.d**" à partir d'un fichier au format "**slapd.conf**" (voir en annexe).

5 On-Line Configuration = manipulable à chaud, sans redémarrer le service slapd.

Notes personnelles

5.2- Olc et le répertoire "slapd.d"

Il contient une configuration dite "**olc**".

un fichier "**cn=config.ldif**" : paramètres généraux du serveur (pid-file, certificat TLS, ...)

un répertoire "**cn=config**" contenant :

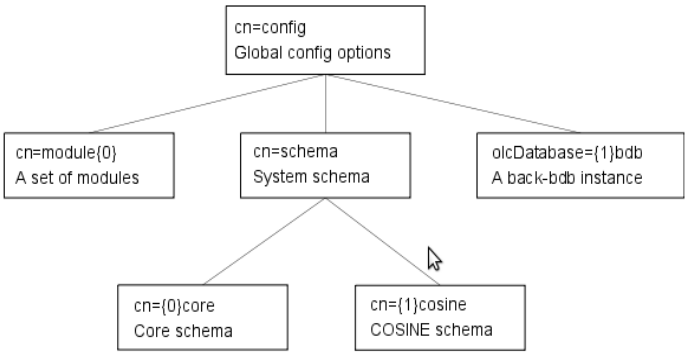
un répertoire "**cn=schema**" définissant les schémas supportés ,

des fichiers "**olcBackend**" de paramétrage des backend de stockage,

des fichiers "**olcDatabase**" de paramétrage des annuaires,

...frontend, ...config, ...monitor...

un fichier du type "...**{n}*db.ldif**" par annuaire hébergé.



On ne touche "jamais" à ces fichiers à la main, mais :

On les produit peut-être initialement par la commande "**slaptest -f… -F ...**"

On les modifie au travers de commandes "**ldapmodify**" ou l'éditeur "**ldapvi**[6]".

6 Attention, "ldapvi" l'option "-H" est remplacée par "-h"… qui indique une URL

Notes personnelles

5.3- Administrer avec la configuration "Olc"

Pour toute modification de la configuration, il faut :
- utiliser la socket locale (unix) : **-H ldapi:///**
- s'authentifier grâce au mécanisme basé sur l'OS : **-Y EXTERNAL**
- spécifier la base "**cn=config**" dans les requêtes de recherche.

5.3.1- Déterminer le DN l'annuaire par défaut : {1} ou {2} ?

Rechercher dans **cn=config** le DN des enregistrements dotés d'une propriété "**olcSuffix**" non nulle :

```
# ldapsearch -QLLL -Y EXTERNAL -H ldapi:/// -b cn=config '(olcSuffix=*)' dn
dn: olcDatabase={2}mdb,cn=config
```

Ici (Stream 8 ou Stream 9), ce sera donc olcDatabase={2}mdb,cn=config.
- Sur des systèmes Ubuntu 22.04 ou Debian 11, ce serait olcDatabase={**1**}mdb,cn=config.
- C'est différent sur CentOS 7 (base HDB) : olcDatabase={2}**hdb**,cn=config

L'écart est du à une base **olcDatabase={1}monitor,cn=config** présente chez CentOS/Stream.

```
# ldapsearch -QLLL -Y EXTERNAL -H ldapi:/// -b cn=config '(objectclass=olcDatabaseConfig)'  dn
```

⇒ La base "Monitor" n'est par défaut pas présente dans le monde Debian. On peut la rajouter.

Notes personnelles

5.4- Les logs du serveur

5.4.1- LogLevel / olcLogLevel

La directive **LogLevel** (slapd.conf) ou **olcLogLevel** (slapd.d) permet de spécifier le niveau de log que l'on souhaite émettre dans **SYSLOG**. En son absence : pas de logs.

Les niveaux sont les suivants :

```
1       (0x1 trace) trace function calls
2       (0x2 packets) debug packet handling
4       (0x4 args) heavy trace debugging (function args)
8       (0x8 conns) connection management
16      (0x10 BER) print out packets sent and received
32      (0x20 filter) search filter processing
64      (0x40 config) configuration file processing
128     (0x80 ACL) access control list processing
256     (0x100 stats) connections, LDAP operations, results (recommended)
512     (0x200 stats2) stats2 log entries sent
1024    (0x400 shell) print communication with shell backends
2048    (0x800 parse) entry parsing

16384   (0x4000 sync) LDAPSync replication
32768   (0x8000 none) only messages that get logged whatever log level is set
```

Autres valeurs :

"**all**" (ou -1) pour tous les niveaux,

"**none**" (ou 0) pour ne loguer que les messages indépendants des niveaux ci dessus.

Niveau par défaut : "stats".

Notes personnelles

5.4.2- Choisir le niveau de logs

On peut indiquer les valeurs : en décimal, en Hexa, ou grâce à leur nom.

Plusieurs niveaux : on additionne, ou on liste chaque valeur ou nom de niveau.

Exemples :

```
olcLogLevel: 129
olcLogLevel: 0x81
olcLogLevel: 128 1
olcLogLevel: 0x80 0x1
olcLogLevel: acl trace
```

5.4.3- Logs de debug du serveur

Les logs des niveaux "packets" (2), "BER" (16), et "parse" (2048) sont uniquement émis vers ***stderr*** et ne sont pas émis vers SYSLOG.

La directive **Logfile** (slapd.conf) ou **olcLogFile** (slapd.d) permet de désigner un fichier pour stocker les messages de debuggage du serveur, en complément de leur émission sur ***stderr***.

5.4.4- Logs et performances du serveur

SYSLOG devrait écrire de manière **asynchrone** les logs de LDAP.

Il faut donc penser à activer le buffer d'écriture dans la configuration de Rsyslog.

Notes personnelles

5.4.5- Choisir le bon niveau de log

Le niveau 0 (rien) n'est pas idéal pour identifier un problème à sa première occurrence...

Le niveau idéal semble 256 (statistiques)
⇒ 1 à 3 lignes de log à chaque requête.

En revanche, il ne permet pas de tracer un problème.
Mais mieux vaut utiliser le flag debug le cas échéant, que de tracer au niveau debug.

Messages importants :
Une application a utilisé un filtre **eq** sur un attribut "foo" (foo=xxxxx) et "foo" n'est pas indexé.

```
"<= bdb_equality_candidates: (foo) index_param failed (18)"
```

Selon la fréquence de ces messages, un index devrait être ajouté.

Notes personnelles

5.5- Lister les annuaires existants

Un annuaire est une "*olcDatabase*" (dont le nom n'est pas vide) : (olcDatabase=*),
qui porte des données héritées d'un Suffixe (**olcSuffix** non vide) : (olcSuffix=*).
son administrateur est défini par **olcRootDN**, dont le mot de passe est défini par **olcRootPW**.

Le listage des annuaires présents peut donc ressembler à :

Sous "CentOS Stream 9" :

```
# ldapsearch -QLLL -Y EXTERNAL -H ldapi:/// -LLL -b cn=config '(&(olcDatabase=*)(olcSuffix=*))'
olcSuffix olcRootDN olcRootPW
dn: olcDatabase={1}mdb,cn=config
olcSuffix: dc=my-domain,dc=com
olcRootDN: cn=Manager,dc=my-domain,dc=com
```

Sous Debian/Ubuntu :

```
# ldapsearch -QLLL -Y EXTERNAL -H ldapi:/// -LLL -b cn=config '(&(olcDatabase=*)(olcSuffix=*))'
olcSuffix olcRootDN olcRootPW
dn: olcDatabase={1}mdb,cn=config
olcSuffix: dc=local
olcRootDN: cn=admin,dc=local
olcRootPW: {SSHA}d8mNYJu+3vPDv7KdEdA/up3mbAcB3TWr
```

Par défaut, ici, il y a un attribut mot de passe que nous choisi à l'installation.

Notes personnelles

5.6- L'administrateur de l'annuaire

Nous supposons dans ce qui suit que l'annuaire par défaut a le DN suivant

olcDatabase={1}mdb,cn=config

Éditons la définition de cet annuaire…

```
#            ldapvi -Y EXTERNAL -h ldapi:/// -b {1}mdb,cn=config
```

5.6.1- "RootDN" : nom de l'administrateur

⇒ modifier le champ **olcRootDN** de l'entrée dont le DN est **olcDatabase={1}mdb,cn=config**

Et son nom, à cet annuaire ?

⇒ modifier la propriété **olcSuffix** de l'entrée dont le DN est **olcDatabase={1}mdb,cn=config**

Attention, sous Debian / Ubuntu, une entrée du nom de la racine (le suffixe) existe par défaut dans l'annuaire. Elle doit être supprimée au préalable, mais il faudra plus tard une entrée racine pour peupler l'annuaire.

Et le mot de passe de l'admin ?

Il faut parfois savoir tourner la page.

Notes personnelles

5.6.2- Changer le mot de passe de "RootDN"

On a intérêt à stocker des mots de passe hachés, et non en clair.

1/ Créer un hach d'un mot de passe avec la commande **slappasswd** :

```
# slappasswd
New password:               (passroot)
Re-enter new password:      (passroot)
{SSHA}+Bj/TnGQJM4+a6Vej2uBRNe7HL7fWzGo
```

2/ Créer un fichier "**chrootpw.ldif**" dont le contenu est le suivant :

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcRootPW
olcRootPW: {SSHA}+Bj/TnGQJM4+a6Vej2uBRNe7HL7fWzGo
```

3/ Exécuter cette requête de modification (**ldapmodify**) de l'annuaire cn=config :

```
# ldapmodify -Q -H ldapi:///   -Y EXTERNAL   -f chrootpw.ldif
modifying entry "olcDatabase={1}mdb,cn=config"
```

Il faut parfois aussi savoir tourner une seconde page…

Notes personnelles

Tout ça avec un seul ldapmodify ?

```
# ldapmodify -Q -H ldapi:///   -Y EXTERNAL
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcsuffix
olcsuffix: dc=example,dc=fr
-
replace: olcrootdn
olcrootdn: cn=manager,dc=example,dc=fr
-
replace: olcrootpw
olcrootpw: {SSHA}bWn23pdsB6At2dH2sZN3L5HxqiV17NpF
<2 fois ENTER>, puis <CTRL-D>
```

Si ce fichier LDIF est intégré…

⇒ l'annuaire a pour racine "dc=example,dc=fr",

⇒ le RDN de son administrateur devient "cn=manager",

⇒ son mot de passe est "123Soleil".

(Debian) cela suppose que l'annuaire soit vidé.

Nous supposons que ce fichier est intégré, et que la racine de l'annuaire (vide) a le DN suivant

dc=example,dc=fr

On peut donc tester l'authentification :

```
# ldapwhoami -D cn=manager,dc=example,dc=fr -w 123Soleil
dn:cn=manager,dc=example,dc=fr
```


Notes personnelles

5.7- Créer un nouvel annuaire

Ci-dessous le listage d'un des annuaires identifiés précédemment :

```
# ldapsearch -Q -Y EXTERNAL -H ldapi:/// -LLL -b olcDatabase={1}mdb,cn=config -o ldif-wrap=no |tee test.ldif
dn: olcDatabase={1}mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: {1}mdb
olcDbDirectory: /var/lib/ldap
olcDbIndex: objectClass eq,pres
olcDbIndex: ou,cn,mail,surname,givenname eq,pres,sub
olcSuffix: dc=example,dc=fr
olcRootDN: cn=manager,dc=example,dc=fr
olcRootPW: {SSHA}bWn23pdsB6At2dH2sZN3L5HxqiV17NpF
```

1/ Éditer "test.ldif", et modifier... **olcSuffix**, **olcDbDirectory**, **olcRootDN** et **olcRootPW**

Ne pas s'ennuyer avec le numéro dans le DN, car l'ajout va "pousser les autres déclarations"

2/ Créer le répertoire désigné par **olcDbDirectory** (attention aux permissions)

!! Ne pas utiliser le même répertoire pour deux annuaires différents !

(CentOS / Stream) : install -d -o ldap -g ldap -m 700 /var/lib/ldap/example.com

3/ Puis insérer les données dans l'annuaire, par **ldapadd**.

```
# ldapadd  -H ldapi:///   -Y EXTERNAL   -f test.ldif
```


Notes personnelles

5.8- Supprimer un annuaire

1/ Supprimer l'entrée décrivant l'annuaire
L'opération de suppression (ldapdelete) dans olc est acceptée en **v2.6** (RHEL 9 / Stream 9 / Alma 9).

```
# ldapdelete  -H ldapi:///   -Y EXTERNAL   "olcDatabase={2}mdb,cn=config"
```

Pour les autres versions, cela se fait serveur arrêté.

```
# systemctl stop slapd
```

Il faut donc supprimer le (bon) fichier ldif. **Exemple** : pour supprimer "dc=example,dc=com"

```
# grep -r example.*com /etc/*ldap/slapd.d/cn\=config/*
/etc/openldap/slapd.d/cn=config/olcDatabase={2}mdb.ldif:olcDbDirectory: /var/lib/ldap/example.com
/etc/openldap/slapd.d/cn=config/olcDatabase={2}mdb.ldif:olcSuffix: dc=example,dc=com
/etc/openldap/slapd.d/cn=config/olcDatabase={2}mdb.ldif:olcRootDN: cn=manager,dc=example,dc=com

# rm -f  /etc/*ldap/slapd.d/cn=config/olcDatabase={2}mdb.ldif
```

2/ Supprimer le répertoire de stockage des données

```
# rm -fr /var/lib/ldap/example.com
```

3/ Redémarrer le serveur

```
# systemctl start slapd
```


Notes personnelles

Au sujet de la suppression de bases :

On peut le faire dynamiquement sur openldap **2.6**.

Exemple ici pour supprimer la base "Monitor"[7], que nous remettrons en service plus tard.

```
# ldapdelete -Q -Y EXTERNAL -H ldapi:/// "olcDatabase={1}monitor,cn=config"
```

⇒ après cette opération, la base de l'annuaire par défaut est renumérotée {1}…

Cette suppression "dynamique" ne fonctionne pas sur Openldap 2.4, pour lequel (cf page précédente) :

on arrête slapd,
depuis /etc/slapd.d/cn=config/, on supprime olcDatabase={1}monitor.ldif,
puis on renomme olcDatabase={2}?db.ldif en ...{1}?db.ldif…
et on redémarre slapd.

7 Cette base donne des informations intéressantes, et on la retrouvera dans le module concernant les backends.

Notes personnelles

5.9- Renommer un annuaire

Sur RedHat / CentOS, l'annuaire par défaut a pour suffixe "dc=my-domain,dc=com".

Exemple, sous CentOS 7, pour modifier ces informations, et ainsi renommer l'annuaire :

1/ Recenser les "DN" à modifier (recherche des lignes "**dn**" des fichiers contenant "my-domain")

```
# grep ^dn  $(grep -rl my-domain /etc/*/slapd.d/cn=config)
/etc/openldap/slapd.d/cn=config/olcDatabase={1}monitor.ldif:dn: olcDatabase={1}monitor
/etc/openldap/slapd.d/cn=config/olcDatabase={2}hdb.ldif:dn: olcDatabase={2}hdb
```

À chacun, il faudra ajouter "**,cn=config**" (voir page suivante)

2/ Déterminer chaque modifications à réaliser dans chaque enregistrement listé

Par exemple, dans le fichier "olcDatabase={2}hdb.ldif", on trouve

olcSuffix: dc=my-domain,dc=com ⇒ base de l'annuaire
olcRootDN: cn=Manager,dc=my-domain,dc=com ⇒ DN de l'admin
olcRootPW : …….. ⇒ Mot de passe de l'admin

3/ Spécifier dans un fichier LDIF les nouvelles valeurs des attributs à modifier.

Opérations LDAP de type "**modify**", réalisant l'action "**replace**" d'un attribut.

4/ Appliquer le fichier LDIF constitué (ldapmodify)

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f chdom1.ldif
```


Notes personnelles

Exemple de fichier de requêtes de modification pour plusieurs éléments : (chdom1.ldif)

dc=my-domain,dc=com		⇒ dc=example,dc=fr
+	cn=Manager	⇒ cn=manager

Fichier "chdom1.ldif" :

```
dn: olcDatabase={1}monitor,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" read by
dn.base="cn=manager,dc=example,dc=fr" read by * none


dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: dc=example,dc=fr


dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: cn=manager,dc=example,dc=fr


dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcRootPW
olcRootPW: {SSHA}f3OgHmIadCeJv5/OQ7qV7RvIbw3XnXWN
```


Notes personnelles

5.10- Importer des schémas

Il s'agit d'apprendre des classes d'objets et attributs au serveur (donc pour tout ses annuaires) :

Dans le format "**slapd.conf**", c'est un "include" d'un fichier ".schema".
Dans le format "**olc**", ce sont des données à insérer dans l'annuaire à partir d'un fichier LDIF.

(dans /etc/ldap/schema, ou /etc/openldap/schema)

```
# ldapadd -Y EXTERNAL -H ldapi:/// -f cosine.ldif
# ldapadd -Y EXTERNAL -H ldapi:/// -f nis.ldif
# ldapadd -Y EXTERNAL -H ldapi:/// -f inetorgperson.ldif
```

Chacun de ces fichiers ".ldif" correspond en fait à un fichier ".schema".

5.10.1- Convertir des fichiers ".schema" au format ".ldif"

Certaines applications peuvent être fournies avec leur schéma… au format "schema".

L'utilitaire "**schema2ldif**" permet de réaliser la conversion.
Package disponible en standard sous Debian,
Pour les autres systèmes: https://github.com/fusiondirectory/schema2ldif

Utilisation : schema2ldif < fichier.schema > fichier.ldif

Notes personnelles

5.11- Stratégie et Format des mots de passe

5.11.1- Les formats de hachage disponibles

Le format de stockage dépend d'un préfixe entre accolades : {MD5}, {SMD5}, {SHA1}, {SSHA}...

Par défaut, le format de stockage est SHA1, réputé cassable depuis 2005.

On peut s'appuyer sur des mécanismes externes pour utiliser d'autres algorithmes :

{SASL} : utiliser la bibliothèque Cyrus SASL

{CRYPT} : utiliser la bibliothèque "crypt" (crypt(5)) de l'OS. (man 5 crypt pour les formats)

Crypt donne accès aux algorithmes dont le code est (comme sur l'OS) encadré par des "$", parmi lesquels : **$1$** : MD5, **$2$** : blowfish, **$5$** : SHA-256, **$6$** : SHA-512, y : yescrypt...

Les formats peuvent être spécifiés au format PHC :

```
$<id>[$<param>=<value>(,<param>=<value>)*][$<salt>[$<hash>]]
```

Cela autorise des paramètres complétant le choix de l'algorithme. SHA-512 possède par exemple un argument implicite "rounds=5000".

Le format correspondant à SHA-512 est en fait :

"**6rounds=5000$%.16s**" et est donc équivalent à "**6%.16s**"

Le " %.16s" en fin de chaîne spécifie la taille du *salt* (16 caractères maxi pour Crypt).

Notes personnelles

5.11.2- Choisir un format

On spécifie le format choisi dans l'entrée **cn=config**[8]

```
# ldapvi -h ldapi:// -Y EXTERNAL -b cn=config
...
olcPasswordHash: {CRYPT}
olcPasswordCryptSaltFormat: $6$%.16s
```

Petite vérification : (suppose l'existence de l'objet "dc=example,dc=fr")

```
# ldappasswd -D cn=manager,dc=example,dc=fr -w 123Soleil -s secret   dc=example,dc=fr
```

```
# ldapsearch -LLL -D cn=manager,dc=example,dc=fr -w 123Soleil -b dc=example,dc=fr -o ldif-wrap=no
userPassword
dn: dc=example,dc=fr
userPassword::
e0NSWVBUfSQ2JGR1NkJxVG1QTHJDb3VqWHckbUNWUGFQVHVFZVI5N3BQNi9jdFQ0SDB1OGFVa0RMcXpxLmJTTmxhaGY0RTFDVkFa
N01LNkUuLzAyb0xDaWlEcE9PYXl0LndJZEk1ek5Mb3VkRXBiTy8=
```

Cela a bien l'air d'être du SHA-512 :

```
# echo
"e0NSWVBUfSQ2JGR1NkJxVG1QTHJDb3VqWHckbUNWUGFQVHVFZVI5N3BQNi9jdFQ0SDB1OGFVa0RMcXpxLmJTTmxhaGY0RTFDVkF
aN01LNkUuLzAyb0xDaWlEcE9PYXl0LndJZEk1ek5Mb3VkRXBiTy8=" | base64 -d

{CRYPT}$6$du6BqTmPLrCoujXw$mCVPaPTuEe… …AZ7MK6E./02oLCiiDpOOayt.wIdI5zNLoudEpbO/
```

8 Dans la version 2.6 (Stream 9) la propriété **olcPasswordHash** doit être déplacée dans le DN "frontend", auquel il faut ajouter l'objectClass **olcFrontEndconfig.** Cela ne peut pas se faire à chaud, mais fonctionne annuaire arrêté par édition du fichier /etc/openldap/slapd.d/cn\=config/olcDatabase\=\{-1\}frontend.ldif,

Notes personnelles

5.12- Stratégies de mots de passe

On peut définir (comme avec shadow), une stratégie de durcissement des mots de passe.

Cela implique :
- de charger le schema ppolicy,
- d'ajouter le module ppolicy,
- d'utiliser un overlay (ppolicy) sur les annuaires concernés,
- d'ajouter une OU et une entrée par stratégie (ici **passwordDefault**).

```
# ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/ppolicy.ldif
# ldapadd -Y EXTERNAL -H ldapi:/// <<DONE
dn: cn=module{0},cn=config
objectClass: olcModuleList
cn: module{0}
olcModuleLoad: ppolicy.la

dn: olcOverlay=ppolicy,olcDatabase={2}mdb,cn=config
objectClass: olcOverlayConfig
objectClass: olcPPolicyConfig
olcOverlay: ppolicy
olcPPolicyDefault: cn=passwordDefault,ou=Policies,dc=domain,dc=local
olcPPolicyHashCleartext: FALSE
olcPPolicyUseLockout: FALSE
olcPPolicyForwardUpdates: FALSE

DONE
```

Ajuster au besoin le DN de l'annuaire concerné (ici olcDatabase={2}mdb,cn=config) et son suffixe (ici dc=domain,dc=local).

Notes personnelles

⇒ L'OU et l'entrée cn=passwordDefault,**ou=Policies,dc=domain,dc=local**

```
# ldapadd -Y EXTERNAL -H ldapi:/// <<DONE
dn: ou=Policies,dc=domain,dc=local
objectClass: organizationalUnit

dn: cn=passwordDefault,ou=Policies,dc=domain,dc=local
objectClass: pwdPolicy
objectClass: person
objectClass: top
cn: passwordDefault
sn: passwordDefault
pwdAttribute: userPassword
pwdCheckQuality: 0
pwdMinAge: 0
pwdMaxAge: 0
pwdMinLength: 8
pwdInHistory: 5
pwdMaxFailure: 3
pwdFailureCountInterval: 0
pwdLockout: TRUE
pwdLockoutDuration: 0
pwdAllowUserChange: TRUE
pwdExpireWarning: 0
pwdGraceAuthNLimit: 0
pwdMustChange: FALSE
pwdSafeModify: FALSE

DONE
```

Documentation des paramètres : ici

Notes personnelles

La stratégie "***passwordDefault***" s'applique alors à toute entrée de type **userPassword**.

Si une entrée doit être paramétrée spécifiquement, il suffit d'indiquer quelle stratégie la concerne :

```
dn: uid=user-special,ou=People,dc=domain,dc=local
changetype: modify
add: pwdPolicySubentry
pwdPolicySubentry: cn=passwordSpecial,ou=Policies,dc=domain,dc=local
```

Documentations :
- man:slapo-ppolicy(5)

- Pour les versions 2.4 & 2.6, la doc officielle est par ici :
  https://openldap.org/doc/admin24/overlays.html#Password%20Policies
  https://openldap.org/doc/admin26/overlays.html#Password%20Policies

- Un "vieux tuto" sur le sujet : https://www.tobru.ch/openldap-password-policy-overlay/

Notes personnelles

6- Administration des bases de données

Notes personnelles

6.1- Utilitaires du paquetage openldap-servers (slapd)

slappasswd : Encrypte un mot de passe

```
# slappasswd
New password:
Re-enter new password:
{SSHA}4A0G0CnFrbgwtT18DfiA6q/61OA4i61a
```

C'est le format à utiliser avec ldapmodify et pour olcRootPW dans la configuration d'un annuaire.

6.1.1- Validation de la configuration

slaptest : teste la validité de la configuration, pouvait être utilisé pour transformer une config "slapd.conf(5)" en config olc ("slapd-config(5)").

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

```
[root@stream9 ~]# slapdn  dc=example,dc=fr
olcPasswordHash: value #0: setting password scheme in the global entry is deprecated. The server may
refuse to start if it is provided by a loadable module, please move it to the frontend database
instead
```

⇒ Dans la version 2.6, la propriété **olcPasswordHash** doit être déplacée dans le DN "frontend"[9].

9 Editer l'objet **olcDatabase={-1}frontend,cn=config**, lui ajouter l'objectClass **olcFrontendConfig**, puis la propriété **olcPasswordHash : {CRYPT}**. Cela ne semble pas pouvoir être fait à chaud, mais fonctionne annuaire arrêté par édition du fichier /etc/openldap/slapd.d/cn\=config/olcDatabase\=\{-1\}frontend.ldif.

Notes personnelles

6.1.2- Maintenance de l'annuaire

Ré-indexation de l'annuaire

slapindex : Indexe à nouveau l'annuaire LDAP à partir du contenu actuel.

```
# slapindex -t -b dc=example,dc=fr[10]
```

(-t) : destruction de la base d'indexes avant réindexation.

Sauvegarde de l'annuaire

slapcat : Exporter des données de l'annuaire LDAP "example.fr" dans un fichier LDIF.

```
# slapcat -l output.ldif -b dc=example,dc=fr
```

⇒ slapindex et slapcat sont utilisables sur un annuaire démarré[11], mais il ne doit pas subir de modification pour garantir l'intégrité des données.

Restauration de l'annuaire

slapadd : Importer des entrées d'un fichier LDIF dans un annuaire LDAP (le 2ème).

```
# slapadd -l input.ldif -n 2
```

⇒ slapadd n'est **à utiliser que lorsque l'annuaire est ARRÊTÉ**

10 Au lieu de "-b" pour spécifier sa racine, on peut utiliser "-n" pour spécifier un numéro d'annuaire ou rien pour tous les annuaires.
11 Plutôt que **slapcat**, la bonne pratique serait plutôt l'utilisation de **ldapsearch**, en intégrant les attributs opérationnels.

Notes personnelles

6.2- Contrôle d'accès aux données

Les **ACLs** permettent de **définir des droits d'accès à l'annuaire**. man slapd.access(5)

Elles citent un "**objet**" : attribut, entrée, ensemble d'entrées (base d'une arborescence,

puis éventuellement citent un "**sujet**" : utilisateur(s), groupe(s), DN, machine (ip, nom.…)

et énoncent l'"**autorisation**" et éventuellement un "**contrôle**"

6.2.1- La directive access ou l'attribut olcAccess

La syntaxe est la même : "**access to…**" avec slapd.conf, et "**olcAccess : to…**" dans slapd.d…

```
access to <what> [ by <who> <access> [ <control> ] ]+
```

```
olcAccess : {0}to <what> [ by <who> <access> [ <control>
```

À savoir :

Par défaut, … access to * by * read

RootDN a tous les droits sur son propre annuaire, même en l'absence d'ACLs.

Notes personnelles

6.2.2- L'ordre des règles

Elles sont **évaluées dans leur ordre de déclaration**.

Dès qu'une ACL correspond à la cible recherchée (to **what**), on vérifie le sujet (by **who**)

Si la clause "**who**" correspond à l'utilisateur demandeur, on évalue **access** et **control**.

Chaque clause "**what**" est implicitement terminée par : **to * by * none**

En l'absence de "**control**", la recherche est arrêtée à la première correspondance **what** + **who**.

Chaque clause "**who**" est implicitement terminée par : **by * none stop**

On déclare donc les ACLS générales **APRÈS** les ACLS précises.

Notes personnelles

6.2.3- L'objet (what)

L'objet peut être :

```
*
dn[.<dnstyle>]=<dnpattern>
filter=<ldapfilter>
attrs=<attrlist>[ val[/matchingRule][.<attrstyle>]=<attrval>]
```

Le <**dnpattern**> peut être un DN d'une entrée
ou simplement "*" (dans ce cas, dn= est optionnel).

Le <**ldapfilter**> par défaut est "(objectClass=*)"

Pour "<**attrlist**>", on peut spécifier une liste d'attributs, "entry", "children", ou "@objectClass"
Liste vide ⇒ tous les attributs sont concernés

Pour <**dnstyle**>, on peut spécifier
base, **sub**, **one**, **children** : idem aux filtres ldapsearch.
regex : dans ce cas, **dnpattern** est une **expression rationnelle** :
concerne alors tout DN qui correspond à la regex.

Exemple…

```
access to dn.regex="^(.+,)?uid=([^,]+),dc=([^,]+),dc=com$"...
     by dn.exact,expand="uid=$2,dc=$3,dc=com" write
```


Notes personnelles

6.2.4- Le sujet (who)

On peut utiliser notamment (voir man slapd.access pour les autres possibilités)

***** : tout utilisateur
anonymous : l'utilisateur anonyme (dont le DN est "")

users : tout utilisateur authentifié
group : le DN d'un groupe d'utilisateurs
Voir man slapd.access pour les détails (/classe/attribut)

self : l'utilisateur correspondant au DN authentifié.
On peut préciser un "level" : self.{-1} = le parent de l'objet (what).
dn : le DN cité.
On peut préciser un "level".

peername : <peer-name>.ip=<ip>[%<mask>][{<n>}]
peer-name.ip=192.168.1.0%255.255.255.0
peername.ip=192.168.1.16%255.255.255.240{9009}

domain : le nom d'hôte (ou une partie) du client
domain.subtree=dom1.com : matche "www.dom1.com" mais pas "www.newdom1.com"
La résolution de nom inverse est désactivée par défaut.
Voir reverse-lookup (olcReverseLookup), qu'il n'est pas conseillé d'activer.

Notes personnelles

6.2.5- L'autorisation (access)

Une des valeurs suivantes :

```
[[real]self]{<level>|<priv>}
<level> ::= none|disclose|auth|compare|search|read|{write|add|delete}|manage
<priv> ::= {=|+|-}{0|d|x|c|s|r|{w|a|z}|m}+
```

realself : accord si l'utilisateur est authentifié
self : accord pour l'utilisateur demandant l'authentification

Level : du moins ouvert au plus ouvert. N+1 engendre forcément N.
- **none** : aucun accès
- **disclose** : affichage des informations d'erreur
- **auth** : pour l'authentification
- **compare** : opérations de comparaisons
- **search** : listages, recherches
- **read** : listage des propriétés
- **write** (**add** / **delete**) : write = add + delete
- **manage** : tous les accès y compris administratifs

Priv : autre méthode : ajout explicite pour chaque clause "access to".
= / + / - : comme pour chmod
0 (none), (d) disclose, (x) auth, (c) compare, (r) search,
(w) write / (a) add / (z) delete / (m) manage...
Par défaut, si aucun accès n'est offert, c'est **+0**

Notes personnelles

6.2.6- Le contrôle (control)

3 possibilités

stop (par défaut) : on s'arrête à la première correspondance what + who

break : passe aux autres clauses "access to" concernant le même objet (what).

continue : on continue à lire la clause "access to"
pour prendre en compte les autres clauses concernant les autres sujets (who)
qui peuvent donc modifier les privilèges accordés (incrémentiel)

6.2.7- Exemples

On autorise l'accès à l'attribut « *userPassword* »
pour l'authentification (**auth**) des utilisateurs non encore authentifiés (anonymous)
on accorde le droit de le modifier (**write**) son propre mot de passe (self)
aucun accès (**none**) pour les autres utilisateurs (*****)

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

On accorde à tout le monde (by *****) le droit de lire tout l'annuaire (to *****)

```
access to *   by * read
```


Notes personnelles

6.2.8- Tester les ACLs

Les commandes "ldap*" sont bien entendu soumises aux ACLs définies dans chaque base.

On peut aussi utiliser **slapacl** :

Cette commande lit les règles "access ..." dans slapd.conf ou "olcAccess… " dans cn=config et permet te tester un type d'accès (ici read) sur un attribut ou plusieurs (ici userpassword) :

```
$ sudo -i
# slapacl -v -D uid=tom,ou=opendoor.fr,dc=od -b uid=sophie,ou=lesconstans.fr,dc=od userpassword/read
read access to userPassword: DENIED
```

En l'absence du couple attribut/valeur, c'est l'entrée de base qui est testée.

Notes personnelles

6.3- Indexation des bases

Les index permettent d'**accélérer les recherches** dans l'annuaire.
S'il en manque : message type "mdb_equality_candidates: (entryUUID) not indexed" dans les log.
Attention, trop d'index tue les performances en écriture !

La directive **index** (**olcDbIndex:**) introduit le nom d'un attribut et un critère qui sera souvent utilisé

```
index           objectClass eq
```

On peut préciser un même critère pour plusieurs attributs, dans ce cas, on factorise le critère :

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

Types d'index :

Index	Type de recherche (filtre)
eq	égalité stricte ('uid=fmicaux')
sub	utilisation d'un wildcard ('uid=f*')
subinitial	optimisation de sub pour wildcard à la fin ('uid=fmic*')
subfinal	optimisation de sub pour wildcard au début ('uid=*aux')
subany	optimisation de sub pour wilcard au début ou à la fin ('uid=*ic*')
approx	recherche par approximation sonore ('uid~=miko')
pres	recherche de présence ('objectClass=posixAccount')

Notes personnelles

6.4- Les backends et les modules

Un **backend** est la plupart du temps le moteur stockage des données de l'annuaire.

LDIF, un backend lent mais facile à mettre en œuvre
LDBM n'est plus fourni, BDB, plus robuste l'avait remplacé.
HDB utilise une structure hiérarchique et permet de renommer des branches de l'arborescence.
MDB travaille en mémoire et est celui conseillé dans les dernières versions de Slapd.

Les backends sont fournis sous forme de modules, stockés (Debian) dans /usr/lib/ldap

```
# cd /usr/lib/ldap
# echo back*.la
back_bdb.la back_dnssrv.la back_hdb.la back_ldap.la back_mdb.la back_meta.la back_monitor.la
back_null.la back_passwd.la back_perl.la back_relay.la back_shell.la back_sock.la back_sql.la
```

6.4.1- Les backends "built-in"

Sur AlmaLinux 9 (v2.6), le module "back_mdb.la" n'est pas fourni, mais le backend est pourtant dispo et c'est celui par défaut... C'est sûrement que ce backend est built-in dès la compilation :

```
# ldapsearch -Q -Y EXTERNAL -H ldapi:/// -LLL -s base -b cn=Backends,cn=Monitor monitoredInfo
dn: cn=Backends,cn=Monitor
monitoredInfo: config
monitoredInfo: ldif
monitoredInfo: monitor
monitoredInfo: mdb
monitoredInfo: perl
```


Notes personnelles

6.4.2- Lister les modules chargés

Pour pouvoir utiliser un backend particulier, **le module** doit être chargé…

```
# ldapsearch -Q -Y EXTERNAL -H ldapi:/// -LLL -b cn=config "(objectClass=olcModuleList)" -o ldif-
wrap=no olcModuleLoad
dn: cn=module{0},cn=config
olcModuleLoad: {0}back_mdb

dn: cn=module{1},cn=config
olcModuleLoad: {0}back_hdb
```

⇒ ici **mdb** et **hdb** sont dispo (chargés)

Chaque backend a ses propres propriétés, notamment ses directives de configuration / tuning.

Documentation : man **slapd.backends**(5) pour leur description.

Voir aussi "**Les backends**" et "**Directives bdb et hdb**"

Notes personnelles

6.4.3- Charger un module : olcModuleLoad

Au départ, sous CentOS/Alma aucun module n'est chargé. Sous Debian, seul "mdb" est chargé.

```
# ldapsearch -Q -Y EXTERNAL -H ldapi:/// -LLL -b cn=config "(objectClass=olcModuleList)" -o ldif-
wrap=no olcModulePath olcModuleLoad
dn: cn=module{0},cn=config
olcModulePath: /usr/lib/ldap
olcModuleLoad: {0}back_mdb
```

Le fichier LDIF suivant permet de charger un module (ici le backend "null") :

```
dn: cn=module,cn=config
objectClass: olcModuleList
cn: module
olcModulePath: /usr/lib64/openldap
olcModuleLoad: back_null
```

Note : le **{numéro d'ordre}** affiché au listage est important et il ne faut pas de doublon. Le mieux est de le supprimer du fichier LDIF et le serveur en ajoutera un tout seul.

Note : sur CentOS / AlmaLinux : le ModulePath est **/usr/lib64/openldap**.

```
# ldapadd  -Q -H ldapi:///   -Y EXTERNAL   -f back_load.ldif
adding new entry "cn=module{1},cn=config"
```

Une erreur ? Vérifier la syntaxe (chemin + nom du module) + vérifier que pas déjà chargé !

```
ldap_add: Other (e.g., implementation specific) error (80)
      additional info: <olcModuleLoad> handler exited with 1

# journalctl -u slapd
janv. 23 19:16:30 debian-csi slapd[3177]: lt_dlopenext failed: (back_nulll) file not found
janv. 23 19:27:35 debian-csi slapd[3177]: module_load: (back_null) already loaded
```


Notes personnelles

6.4.4- Activer un backend : olcBackend...

Le chargement du module ne suffit pas, il faut aussi que le backend voulu soit déclaré :

Fichier back_hdb.ldif :

```
# Activation du backend
dn: olcBackend={1}hdb,cn=config
objectClass: olcBackendConfig
olcBackend: {1}hdb
```

Il faut ensuite charger ce fichier LDIF classiquement…

```
ldapadd -Y EXTERNAL -H ldapi:/// -f back.ldif
```

Notes concernant les backends sur RedHat :

Les backends peuvent être compilés en "dur" dans slapd.

RedHat semble avoir fait ce choix, au moins pour Mdb et Hdb, qui sont donc tous deux disponibles sans avoir à les charger explicitement. En version 2.6 (RedHat 9), HDB n'est plus fourni.

Notes personnelles

6.4.5- Créer un annuaire utilisant un backend spécifique

1/ Pour utiliser le moteur Hdb, on crée un fichier LDIF, similaire à ceux (Mdb) déjà rencontrés,
⇒ On remplacera les occurrences de "Mdb" par "Hdb"

```
# cat basehdb.ldif
dn: olcDatabase={1}hdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcHdbConfig
olcDatabase: {1}hdb
olcDbDirectory: /var/lib/ldaphdb
olcSuffix: dc=example,dc=fr
olcAccess: {0}to attrs=userPassword,shadowLastChange by self write by anonymous auth by * none
olcAccess: {1}to dn.base="" by * read
olcAccess: {2}to * by * read
olcLastMod: TRUE
olcDbIndex: objectClass eq
olcDbIndex: cn,uid eq
olcDbIndex: uidNumber,gidNumber eq
olcDbIndex: member,memberUid eq
olcDbCheckpoint: 512 30
olcDbConfig: set_lg_bsize 2097512
olcDbConfig: set_cachesize 0 10485760 0
olcDbConfig: set_flags DB_LOG_AUTOREMOVE
olcDbCacheSize: 20000
olcDbIDLcacheSize: 320000
olcRootDN: cn=root,dc=example,dc=fr
olcRootPW: {SSHA}+Bj/TnGQJM4+a6Vej2uBRNe7HL7fWzGo
```

2/ On crée le répertoire "/var/lib/ldaphdb" (avec les bons droits)

3/ On importe (ldapadd…)

Notes personnelles

4/ Après import de cette "nouvelle base", le répertoire /var/lib/ldaphdb contient ceci :

```
# ls -l /var/lib/ldaphdb/
total 1116
-rw-r--r-- 1 openldap openldap     4096 mai   24 23:52 alock
-rw------- 1 openldap openldap  2326527 mai   24 23:52 __db.001
-rw------- 1 openldap openldap   729087 mai   24 23:52 __db.002
-rw------- 1 openldap openldap   163839 mai   24 23:52 __db.003
-rw-r--r-- 1 openldap openldap       76 mai   24 23:52 DB_CONFIG
-rw------- 1 openldap openldap     8192 mai   24 23:52 dn2id.bdb
-rw------- 1 openldap openldap    32768 mai   24 23:52 id2entry.bdb
-rw------- 1 openldap openldap 10485759 mai   24 23:52 log.0000000001
```

On retrouve ceci dans le fichier DB_CONFIG[12] :

```
# cat /var/lib/ldaphdb/DB_CONFIG
set_lg_bsize 2097512
set_cachesize 0 10485760 0
set_flags DB_LOG_AUTOREMOVE
```

Il a été créé à la volée grâce à la classe olcDatabaseConfig, et aux directives de paramétrage présentes dans le fichier LDIF :

```
olcDbConfig: set_lg_bsize 2097512
olcDbConfig: set_cachesize 0 10485760 0
olcDbConfig: set_flags DB_LOG_AUTOREMOVE
```

Documentation : **slapd-mdb**(5), **slapd-hdb**(5), … **DB_CONFIG**...**DB_CONFIG_set_flags**...

12 Voir : http://sepp.oetiker.ch/db-4.2.52-mo/ref/env/intro.html ou https://docs.oracle.com/cd/E17076_02/html/api_reference/C/env.html

Notes personnelles

6.5- Le backend Monitor

6.5.1- Le backend Monitor

Il offre des possibilités de monitoring.

On active la base monitor :

```
# {2}monitor, config
dn: olcDatabase={2}monitor,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {2}monitor
olcAccess: {0}to *  by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" read  by dn
.base="cn=manager,dc=example,dc=fr" read  by * none
olcAddContentAcl: FALSE
olcLastMod: TRUE
olcMaxDerefDepth: 15
olcReadOnly: FALSE
olcSyncUseSubentry: FALSE
olcMonitoring: FALSE
```

Sur chaque base (database) l'attribut "olcMonitoring :" indique si elle est monitorée.

```
dn: olcDatabase={3}hdb,cn=config
olcMonitoring: TRUE
```

L'arborescence est dans **cn=Monitor** et les informations sont souvent dans les attributs opérationnels.

```
# ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=Monitor   +
```

Voir : http://www.openldap.org/doc/admin26/monitoringslapd.html.

Notes personnelles

6.6- Tuning

6.6.1- Facteurs de performance

Mémoire : la mémoire inutilisée ne sert à rien

⇒ Le cache devrait utiliser au mieux la mémoire disponible.

⇒ Augmenter la RAM pour pouvoir augmenter le cache si possible.

Cache : http://www.openldap.org/doc/admin26/tuning.html#Caching

Stockage : répartir les données sur plusieurs filesystems

⇒ Filesystem : ext2 ou jfs sont réputés plus rapides en écriture.

⇒ Si plusieurs annuaires, chacun son filesystem

⇒ avec BDB/HDB : Un filesystem pour les data, un autre pour les logs

olcDbConfig: **set_lg_dir** /var/tmp/bdb-log

Processeur / Threads : en général, 4 threads par CPU core.

Si on augmente le nombre de threads, les performances en lecture baissent,

Mais les performances en écriture s'améliorent (sous forte charge d'écriture).

Ne pas dépasser 16 threads pour de bonnes performances en lecture.

Les indexes : en fonction de l'usage de l'annuaire

Un filtre de recherche qui n'est pas indexé engendre un parcours de tout l'annuaire.

Chaque index présent doit être maintenu par le serveur, et coute en écriture.

On crée des index de type "**eq**" sur les attributs cn, sn, givenname, mail…

Les index de type "**pres**" sont plutôt peu utilisés, donc engendrent un cout inutile.

Notes personnelles

6.6.2- Les réglages du cache (HDB)

Berkeley DB Cache : deux approches pour le réglage (attribut **olcDbCacheSize**:)

1/ Soit on vise à optimiser le temps d'un slapadd.
La taille optimale est : le volume occupé par les fichiers "*.bdb" de l'annuaire

2/ Soit on vise de bonnes performances une fois les données chargées.
La taille optimale est : celle de id2entry.bdb + 10 %

On doit faire ce réglage dans chaque base BDB ou HDB.
Voir : http://www.openldap.org/doc/admin24/tuning.html#Caching

Le "Entry Cache" de slapd : cachesize (ou olcCacheSize :)… slapd-bdb(5)

Nombre d'entrées décodées pouvant être dans le cache de Slapd.

Le cache IDL : le cache d'Index (attribut **olcDbIDLcacheSize**:)… slapd-bdb(5)

Ce paramètre détermine le nombre d'IDL acceptés.
Chaque IDL mémorise le résultat d'une requête donnée.
Plus il et élevé, meilleures sont les performances des recherches fréquentes sur les mêmes index
Pour BDB : la taille de l'Entry-Cache
Pour HDB : 3 * la taille de l'Entry Cache est une bonne valeur.

Notes personnelles

6.6.3- Les réglages du cache (LMDB)

Il n'y a pas de cache à gérer avec ce backend.

Le principal paramètre de perf pour LMDB est **olcMaxSize** (bytes) qui est la taille max que la base de données est sensé atteindre.

LMDB s'appuie avant tout sur le cache de l'OS ; On peut donc jouer avec **sysctl** sur les paramètres du noyau **vm.dirty_background_ratio** et **vm.dirty_ratio** (avec des valeurs respectivement à 50 et 80)

6.6.4- Performances en écriture

Ecritures asynchrones (write-back) :
Au risque de perdre en intégrité des données, on peut mettre en œuvre un write-back :

Par l'attribut **olcDbNoSync** : TRUE
ou
Par le flag "**set_flags DB_TXN_NOSYNC**" dans DB_CONFIG (ou via **olcDbConfig**:)

6.6.5- Outils de Benchmark

Slamd : http://dl.thezonemanager.com/slamd/

À lire : http://www.openldap.org/pub/hyc/mdm-paper.pdf

Notes personnelles

7- TLS, chiffrement des échanges

Notes personnelles

7.1- Certificat pour Slapd

On a besoin :
- du certificat de la CA,
- de la clé privée du serveur, et du certificat du serveur signé par le CA.

Les documents sont à ranger dans le sous dossier "**certs**" de /etc/openldap (RedHat / CentOS) ou /etc/ldap (Debian).

7.1.1- Création du CA

C'est un certificat auto-signé "qui peut signer" (basicConstraints = CA:TRUE[13])

Documents en sortie : ca.pem & ca-key.pem

```
# install -d -m 2750 -o root -g ldap /etc/openldap/certs
# openssl req \
  -newkey rsa:4096 -nodes -keyout /etc/openldap/certs/ca-key.pem \
  -subj '/countryName=FR/stateOrProvinceName=Bretagne/organizationName=Actilis/CN=CA-LDAP/'  \
  -x509 -sha256 -days 365  \
  -extensions v3_ca \
  -out /etc/openldap/certs/ca.pem
# chmod 444 /etc/openldap/certs/ca.pem
```

13 Cette "contrainte" est définie dans la section présentant l'extension "v3_ca" du fichier de configuration openssl.cnf.

Notes personnelles

7.1.2- Création d'un certificat serveur

C'est un certificat "normal", signé par le CA.

Documents en sortie : server.csr & key.pem

```
# openssl req \
  -newkey rsa:4096 -nodes -keyout /etc/openldap/certs/key.pem \
  -subj '/CN=ldap.example.fr/'  \
  -out server.csr
# chmod 440 /etc/openldap/certs/key.pem
```

Signature du certificat par le CA (avec les extensions adéquates)

Document en sortie : cert.pem

```
# openssl x509 -req -days 365 -sha256 -in server.csr -out /etc/openldap/certs/cert.pem \
  -CA /etc/openldap/certs/ca.pem -CAkey /etc/openldap/certs/ca-key.pem  -CAcreateserial \
  -extfile <(printf "basicConstraints = CA:FALSE\n subjectAltName=IP:127.0.0.1,DNS:ldap.example.fr")
```

Éventuellement : suppression de "server.csr".

Notes personnelles

7.2- Configurer Slapd pour TLS

La suite suppose que ldapserver.crt[14] et ldapserver.key sont dans /etc/*ldap/certs/[15], et sont lisibles par l'utilisateur "ldap".

1/ Préparer un fichier LDIF : (tls.ldif) pour ajouter (**en une seule opération**) ces 3 attributs.

```
dn: cn=config
changetype: modify
replace: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/openldap/certs/ca.pem
-
replace: olcTLSCertificateFile
olcTLSCertificateFile: /etc/openldap/certs/cert.pem
-
replace: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/openldap/certs/key.pem
```

2/ Importer ce fichier

```
# ldapmodify -Y EXTERNAL -H ldapi:///  < tls.ldif
```

3/ Vérifier que le serveur écoute sur la socket ldaps:/// (prévu par défaut sur CentOS / RedHat ≥ 8)[16]

```
# lsof -i TCP:636
```

4/ Penser au firewall (LDAPS ⇒ tcp/636) le cas échéant,

14 Il s'agit d'un certificat autosigné. Le certificat de l'AC est donc le même que celui du serveur.
15 Sur un système CentOS / RedHat, c'est dans /etc/openldap/certs qu'il faut les ranger, et sous Debian, dans /etc/ldap/certs.
16 Mais pas sur Debian, pour laquelle il faut ajouter "**ldaps:///**" à la variable **SLAPD_SERVICES** dans /etc/default/slapd.

Notes personnelles

7.3- Configurer les clients pour TLS

Dans les fichiers **/etc/openldap/ldap.conf** (**/etc/ldap/ldap.conf**) ajouter les déclarations suivantes :

```
HOST            ldap.example.fr
BASE            dc=example,dc=fr

URI             ldaps://ldap.example.fr

TLS_CACERT      /etc/openldap/certs/ca.pem
TLS_REQCERT     demand
```

On utilise ici **TLS_CACERT**, qui présente le fichier contenant peut-être plusieurs certificats, dont celui du CA signataire des certificats de serveurs contactés[17].

TLS_REQCERT spécifie le contrôle effectué par le client sur le certificat du serveur contacté :

never : pas de demande ni de contrôle du certificat serveur

allow : certificat demandé, si aucun n'est présenté, on continue.
Si un certificat est présenté et non valide, il est ignoré et on continue.

try : certificat demandé, si aucun n'est présenté, on continue.
Si un certificat est présenté et non valide, on abandonne.

demand : (défaut), on exige la présentation d'un certificat valide, sinon on abandonne.

17 Cette configuration est à réaliser sur les **CLIENTS**. Il leur faut bien le certificat du CA pour qu'ils reconnaissent l'autorité. L'idéal reste d'utiliser un "vrai" certificat, cela évite de distribuer le fichier "ca.pem" aux clients.

Notes personnelles

7.3.1- Tester…

Tester la liaison du client LDAP classique : le serveur est contacté sur le port 636 et non plus 389.

```
# ldapwhoami -h 192.168.157.108 -Z   -D "cn=manager,dc=example,dc=fr" -w 123Soleil
ldap_start_tls: Connect error (-11)
       additional info: TLS: hostname does not match subjectAltName in peer certificate
ldap_result: Can't contact LDAP server (-1)
```

On a contacté le serveur par son adresse IP, et ce "nom là" ne fait pas partie de ceux acceptés dans le certificat. Le CN du certificat est **ldap.example.fr**…et il n'est pas valide pour 192.168.157.108.

Les noms valides sont "127.0.0.1" et "ldap.example.fr" (à ajouter dans /etc/hosts ou dans le DNS).

Notez la différence[18] entre :

```
# ldapwhoami -H ldap://127.0.0.1 -ZZ -D "cn=manager,dc=example,dc=fr" -w 123Soleil
dn:cn=manager,dc=example,dc=fr
```

et

```
# ldapwhoami -H ldaps://127.0.0.1 -D "cn=manager,dc=example,dc=fr" -w 123Soleil
dn:cn=manager,dc=example,dc=fr
```

Enfin, avec le nom "ldap.example.fr" :

```
# ldapwhoami -H ldaps://ldap.example.fr -D "cn=manager,dc=example,dc=fr" -w 123Soleil
dn:cn=manager,dc=example,dc=fr
```

18 Dans le premier cas, contact par le port 389 (ldap), avec STARTTLS (et exigence d'un certificat correct : ZZ), dans le second cas, la même exigence, mais contact par le port 636 (ldaps)

Notes personnelles

7.4- Compléments sur TLS/LDAP

Attention du nom du serveur...

Le nom de serveur (**CN**) à saisir lors de la création du certificat doit correspondre au nom de la machine pour laquelle est fait le certificat.

Lorsque l'on résout ce nom, il doit pointer vers l'adresse IP correspondante.
La résolution inverse de l'adresse IP doit pointer vers le nom officiel du serveur.

Confusion dans les certificats

On peut travailler avec un certificat auto-signé.
Dans le cas contraire, ne pas confondre celui du CA et celui du serveur contacté.

Plusieurs CA :

Le fichier défini par **TLS_CACERT** doit contenir TOUS les certificats de TOUTES les autorités de certification (concaténation).

On peut aussi utiliser la directive **TLS_CACERTDIR** pour référencer le répertoire dans lequel on stocke les certificats des CA ayant signé les certificats des serveurs contactés.

Certificat Client

On peut utiliser (uniquement en option utilisateur, dans $HOME/.ldaprc) les directives TLS_KEY et TLS_CERT. Elles présentent un certificat CLIENT, pour **authentifier le client** auprès du serveur.

Notes personnelles

8- Serveur d'authentification LDAP

Notes personnelles

8.1- LDAP et Authentification des utilisateurs

8.1.1- Avant de commencer

Authentifier les comptes utilisateurs via LDAP permet de centraliser la gestion des comptes utilisateur et leurs mots de passe, des groupes, etc...

Les informations concernant les comptes systèmes (root, …) doivent cependant rester locales :

Pour ne pas bloquer l'accès au système en cas de panne du serveur LDAP,
Pour des facilités de maintenance de ces comptes.

Changements en exploitation

Gestion des comptes LOCAUX par les commandes classiques (useradd, groupadd, passwd, …),

Gestion des comptes LDAP via des fichiers LDIF, ou via une application web :

⇒ **LDAP Account Manager** (https://github.com/LDAPAccountManager/lam)[19].
⇒ **Gosa²** (https://github.com/gosa-project)[20]

Sécurité

Les informations concernant les login/password des utilisateurs ne doivent pas circuler en clair, la communication entre les clients et le serveur LDAP doit donc être chiffrée.

19 package "ldap-account-manager" sous Debian.
20 package "gosa" sous Debian. Le projet semble très peu actif. Mort ?

Notes personnelles

8.2- Un annuaire pour gérer des comptes

8.2.1- Le DIT

Les classes adaptées à la gestion des comptes Unix sont documentées dans la RFC 2307,

Elles sont implémentées par le schéma "nis.schema".

posixAccount : implémentation des données de /etc/passwd
shadowAccount : implémentation des extensions de /etc/shadow
posixGroup : gestion des groupes Unix et de leurs membres (/etc/group)

La classe **posixAccount** n'est pas de type "STRUCTURAL",

⇒ une entrée qui l'utilise doit dont s'appuyer sur une autre classe étant "STRUCTURAL"
⇒la classe **account**, répond au besoin, mais ne comporte pas de notion relative au "mail"

La classe **inetOrgPerson** est de type "STRUCTURAL".
⇒ Elle implémente (entre-autre) l'attribut "mail",
⇒ Ses attributs sont optionnels, mais elle hérite de "person", qui implique les attributs **sn** et **cn**.
⇒ Elle est STRUCTURAL : inetOrgPerson + posixAccount = la solution la plus ouverte.

Bonne pratique :
L'ensemble inetOrgPerson + posixAccount permet de gérer des utilisateurs unix ayant une boite mail.

Notes personnelles

8.2.2- Les données

Avec ce type de DIT, on peut gérer des utilisateurs Unix, et des boites mail… pour plusieurs domaines.

```
# ldapsearch  -b dc=example,dc=fr -s children -x -LLL
dn: ou=dom56.local,dc=example,dc=fr
objectClass: organizationalUnit
ou: dom56.local

dn: ou=People,ou=dom56.local,dc=example,dc=fr
objectClass: organizationalUnit
ou: People

dn: ou=Group,ou=dom56.local,dc=example,dc=fr
objectClass: organizationalUnit
ou: Group

dn: cn=dom56users,ou=Group,ou=dom56.local,dc=example,dc=fr
objectClass: posixGroup
cn: dom56users
gidNumber: 1000

dn: uid=usera,ou=People,ou=dom56.local,dc=example,dc=fr
objectClass: inetOrgPerson
objectClass: posixAccount
cn: usera
sn: Utilisateur A
mail: usera@dom56.local
uid: usera
userPassword:: c2VjcmV0YQ==
uidNumber: 10001
gidNumber: 1000
homeDirectory: /home/virtual/usera
```


Notes personnelles

8.3- Bibliothèque libnss_ldap

Histoire des bases de données d'informations système

Traditionnellement, les bases de données d'informations système sont stockées dans des fichiers plats (/etc/passwd, /etc/group, /etc/hosts,...). L'apparition de services de centralisation d'informations (NIS, le DNS) a nécessité au début la ré-écriture des outils (login, su, passwd, ping...) pour prendre en compte ces bases de données...

Les bibliothèques libnss* et le fichier /etc/nsswitch.conf

Devant la multiplication des systèmes de centralisation, à partir de Solaris 2 (et de la version glibc 2.x), un nouveau système a été développé pour la recherche des informations.

Celui-ci est « propre » et paramétrable, il s'appelle « Name Service Switch » (NSS). Il est configuré par le fichier **/etc/nsswitch.conf** (man nsswitch.conf (5))

Le fichier **nsswitch.conf** « prêt pour les services de noms LDAP ».

```
# $ grep -vE '^#|^ *$' /etc/nsswitch.conf
…
passwd:      files ldap
shadow:      files ldap
group:       files ldap
…
hosts:       files dns
…
```

Pour les questions sur **passwd, shadow et group**, si on n'obtient pas la réponse dans le fichier plat , une requête LDAP sera faite, reste à dire vers quel serveur, dans quel annuaire, branche...

Notes personnelles

Le système repose sur plusieurs bibliothèques, chacune consacrée à une source d'information :

Bibliothèque	**Source désignée**
/lib/libnss_compat.so	« compat » (= files avec les « + » en fin de fichier => nis)
/lib/libnss_dns.so	« dns » (uniquement pour noms d'hôtes)
/lib/libnss_files.so	méthode qui se rapporte aux fichiers plats de /etc
/lib/libnss_hesiod.so	« hesoid », lié au DNS (champs « TXT »)
/lib/libnss_ldap.so	méthode LDAP
/lib/libnss_nis.so	méthode NIS (v2)
/lib/libnss_nisplus.so	méthode NISPLUS (v3)
/lib/libnss_winbind.so	méthode SMB (client d'un serveur Windows)
/lib/libnss_wins.so	méthode « WINS » (serveur de noms d'hôtes Windows)

La bibliothèque **libnss_files** repose sur le fichier texte de /etc portant le nom du service demandé. La méthode **files** dans une recherche sur **passwd:** se reporte donc à /etc/passwd...

D'autres bibliothèque ont leur fichier de configuration (libnss_dns => /etc/resolv.conf).

Sous Debian, la bibliothèque **libnss_ldap** utilise le fichier **/etc/libnss-ldap.conf**, et pour **libnss_ldapd**, c'est dans **/etc/nslcd.conf** que les choses se jouent.

Notes personnelles

Configuration (libnss-ldap.conf ou nslcd.conf)

Pour activer TLS du côté client, le fichier de certificat du serveur LDAPS contacté doit être copié (...sans la clé privée...) sur chaque client dans le répertoire référencé par **tls_cacertdir.**
Déclarer alors ceci :

```
host            nom-du-serveur
base            dc=example,dc=fr
ldap_version     3
ssl on
pam_password md5
tls_checkpeer yes
tls_cacertdir /etc/ldap/certs
```

On peut indiquer où et comment chercher les informations (**nss_ldap**)

```
…
binddn dc=example,dc=fr
bindpw passwwwoorrd
…
rootbinddn cn=manager,dc=example,dc=fr
…
timelimit 30
bind_timelimit 5
idle_timelimit 3600
…
…
nss_base_passwd         ou=users,dc=example,dc=fr?one
nss_base_shadow         ou=users,dc=example,dc=fr?one
nss_base_group          ou=groups,dc=example,dc=fr?one
```

DN utilisé si les connexions anonymes sont interdites

DN quand c'est root le demandeur.
Le password est en clair dans /etc/ldap.secrets (600)

Petit tuning

On s'adapte à notre annuaire

Notes personnelles

8.4- Bibliothèque pam_ldap

Les bibliothèques « **nss** » ne sont pas destinées à l'authentification mais à la résolution de noms.

C'est le module de PAM **pam_ldap** qui permet aux applications fonctionnant avec PAM (Authentification système, mail (pop, imap), FTP, Samba...) d'authentifier les utilisateurs en utilisant un serveur LDAP.

Les deux bibliothèques (**pam_ldap** et **nss_ldap**), nécessaires pour l'authentification système, sont fournies par le paquetage **nss_ldap**. Le paquetage installe aussi le fichier de configuration **/etc/ldap.conf** spécifique à ces bibliothèques, mais ne paramètre pas PAM pour qu'il fasse appel au module **pam_ldap**.

Sous GNU/Linux, ce sont les bibliothèques PAM qui réalisent les opérations dans 4 contextes :
(**auth**) authentification, (**account**) contrôle d'accès, (**session**) paramétrage des sessions,
(**password**) modification des informations liées à l'authentification

Faire appel à pam_ldap (exemple Ubuntu)

Le module **pam_ldap** doit être activé dans les 4 contextes où l'on fait déjà appel au module d'authentification à plat (**pam_unix**). Pour « **auth** », le fichier /etc/pam.d/system-auth (RH) ou /etc/pam.d/common-auth (Debian, Ubuntu) pourrait ressembler à ceci :

```
auth    [success=2 default=ignore]      pam_unix.so nullok_secure
auth    [success=1 default=ignore]      pam_ldap.so use_first_pass
auth    requisite                       pam_deny.so
auth    required                        pam_permit.so
```


Notes personnelles

8.5- SSSD : System Security Services Daemon

Versions et docs de référence :

RHEL/CentOS 7 (sssd v1.16,x),
RHEL/Stream 8 (sssd v2.7.x), même version sur Stream 9.
La démarche est facilement adaptable au monde Debian (version 2.8 sur Debian 12).

8.5.1- Installation des packages

Pour certaines tâches, sssd s'appuie sur "Oddjobd", comme par exemple lorsque l'on souhaite automatiser la création du répertoire HOME à la première connexion (man pam_mkhomedir).

En complément de "sssd" et de "sssd-ldap", il faut donc installer "oddjob" et "oddjob-mkhomedir".

```
# dnf -y install sssd sssd-ldap sssd-tools oddjob oddjob-mkhomedir
```

Activer le service **oddjob :**

```
# systemctl enable --now oddjobd.service
```


Notes personnelles

8.5.2- Authselect : configuration du profil d'authentification

En V2 (Stream >=8), **authselect** permet de définir / manipuler / choisir une source d'authentification : minimal (local), domaine (winbind), ldap via sss…

```
# authselect list
- minimal      Local users only for minimal installations
- nis          Enable NIS for system authentication
- sssd         Enable SSSD for system authentication (also for local users only)
- winbind      Enable winbind for system authentication

# authselect current
Profile ID: sssd
Enabled features: None
```

Un backup de l'ancienne configuration a été créé lors d'un changement de profil.

```
# authselect backup-list
2022-10-21-07-38-26.P5ysmn (créé à ven. 21 oct. 2022 09:38:26 CEST)
2022-10-21-07-40-16.P8Vbdc (créé à ven. 21 oct. 2022 09:40:16 CEST)
```

Pour restaurer une configuration précédente :

```
# authselect backup-restore le-nom-du-backup
```


Notes personnelles

Choix d'un profil :

On utilise "**--force**",car des fichiers déjà présents doivent être écrasés :

```
# authselect select --force sssd with-faillock with-mkhomedir
Sauvegarde stockée à /var/lib/authselect/backups/2022-10-21-07-38-26.P5ysmn
Le profil « sssd » a été sélectionné.
Les mappages nsswitch suivants ont été remplacés par le profil :
- passwd
- group
- netgroup
- automount
- services

Make sure that SSSD service is configured and enabled. See SSSD documentation for more information.

- with-mkhomedir is selected, make sure pam_oddjob_mkhomedir module
  is present and oddjobd service is enabled and active
- systemctl enable --now oddjobd.service
```

Fichiers impactés :
NSS : vérifier dans /etc/nsswitch.conf[21].
PAM a aussi été configuré :

```
# grep mkhomedir /etc/pam.d/*
/etc/pam.d/password-auth:session     optional                                  pam_oddjob_mkhomedir.so
/etc/pam.d/system-auth:session     optional                                  pam_oddjob_mkhomedir.so
```

21 Le bibliothèque **sss** est activée à l'installation du package **sssd** pour les services passwd, group, shadow, netgroup…

Notes personnelles

8.5.3- Configuration de sss

Prérequis:

1/ (serveur) LDAP accessible en SSL/TLS.

2/ (serveur) schémas cosine, nis, et inetorgperson chargés

```
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/cosine.ldif
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/nis.ldif
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/inetorgperson.ldif
```

3/ Annuaire peuplé avec à minima ce genre de chose : (example.fr.ldif)

```
dn:dc=example,dc=fr
objectClass: organization
objectClass: dcObject
o: Example Company

dn: ou=People,dc=example,dc=fr
objectClass: organizationalUnit

dn: ou=Group,dc=example,dc=fr
objectClass: organizationalUnit

dn: cn=ldapgroup1,ou=Group,dc=example,dc=fr
objectClass: posixGroup
gidNumber: 10001
```

```
dn: uid=ldapusera,ou=People,dc=example,dc=fr
objectClass: posixAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
cn: ldap usera
sn: ldapusera
uid: ldapusera
uidNumber: 10001
gidNumber: 10001
homeDirectory: /home/ldapusera
userPassword : passa
loginShell: /bin/bash
```

4/ (Clients) BASE et URI configurées dans /etc/openldap/ldap.conf.

Notes personnelles

Fichier de configuration : /etc/sssd/sssd.conf (il **doit être** en mode 600)

```
[sssd]
services = nss, pam
domains = example.fr

[nss]
filter_users = root
filter_groups = root

[domain/example.fr]
cache_credentials = true
id_provider = ldap
auth_provider = ldap

ldap_uri = ldaps://ldap.example.fr
ldap_tls_reqcert = demand
ldap_search_base = dc=example,dc=fr

[pam]
offline_credentials_expiration = 1
offline_failed_login_attempts = 3
offline_failed_login_delay = 5
```

Documentation : pour cette partie : man sssd-ldap(5).

En cas de **problème au démarrage** : /usr/sbin/sssd -i -d 1 ⇒ le log est parlant sur les erreurs de typo.

Notes personnelles

Tester...

```
# sssctl user-checks ldapusera
user: ldapusera
action: acct
service: system-auth

SSSD nss user lookup result:
 - user name: ldapusera
 - user id: 10001
 - group id: 10001
 - gecos: ldap usera
 - home directory: /home/ldapusera
 - shell: /bin/bash

SSSD InfoPipe user lookup result:
 - name: ldapusera
 - uidNumber: 10001
 - gidNumber: 10001
 - gecos: ldap usera
 - homeDirectory: /home/ldapusera
 - loginShell: /bin/bash

testing pam_acct_mgmt

pam_acct_mgmt: Success

PAM Environment:
- no env -
```

```
# su - ldapusera -c "pwd"
/home/ldapusera
```

```
# ssh ldapusera@localhost id -un
ldapusera@localhost's password: passa
ldapusera
```


Notes personnelles

8.6- Libuser : gérer les utilisateurs et groupes

Le paquetage **libuser** peut être utilisé.

Editer le fichier **/etc/libuser.conf** et paramétrer les valeurs (indiquées ici en gras) :

```
…

[defaults]
crypt_style = sha512
modules = ldap
create_modules = ldap

…


[ldap]
server = ldap.example.fr
basedn = dc=example,dc=fr
userBranch = ou=People
groupBranch = ou=Group
binddn = cn=manager,dc=example,dc=fr
…
```

Commandes de gestion des utilisateurs et groupes :

lpasswd,
luseradd, lusermod, luserdel,
lgroupadd, lgroupmod, lgroupdel,

Notes personnelles

9- Architecture maître/esclave et réplication

Notes personnelles

9.1- Réplication d'annuaire : concepts

9.1.1- Objectifs...

Éviter le single point of failure.

Répartir la charge et le trafic réseau
⇒ Conserver de bonnes performances même avec un grand nombre de sollicitations

9.1.2- Roadmap de mise en œuvre

Pré-requis : Les deux annuaires existent, les horloges des machines sont synchronisées.

Roadmap : Mise en place du maître (le **provider**)
Ajout des indexes propres à la réplication,
Chargement et configuration du module *syncprovider,*
Création d'un compte de réplication

Mise en place du client (le **consumer**)
Ajout d'index,
Ajout d'une ou plusieurs entrées "**SyncRepl**", chacune son identifiant (RID).

Refaire les index.

Redémarrer le Consumer.

Notes personnelles

9.1.3- Fonctionnement de la syncrepl

Le moteur **Syncrepl** met en jeu un "**SyncProvider**" (le maître) et un **Consumer** (le client).

SyncProvider est un **overlay** disponible sur tout backend supportant **entryCSN** et **entryUUID**.

Chaque entrée de l'annuaire possède un **entryUUID** et un **entryCSN**.
entryUUID : un identifiant unique pour chaque entrée de l'annuaire (utilisé par le journal)
entryCSN : **C**hange **S**equence **N**umber (voir RFC 4533 : Content Synchronisation Protocol)

Un attribut **contextCSN** est géré en mémoire pour chaque base sur le provider
et écrit sur le disque à l'arrêt du serveur ou à chaque checkpoint.

Le provider utilise **contextCSN** pour déterminer les opérations de synchronisation à réaliser :
ce sont celles dont **l'entryCSN** est supérieur à **contextCSN**.

Au démarrage, si **contextCSN** n'est pas lisible, il faut parcourir toute la base.
En effet, il doit contenir la valeur de la plus grande **entryCSN**.

La doc : https://www.openldap.org/doc/admin26/replication.html

Notes personnelles

9.2- Mise en œuvre de la réplication syncrepl

9.2.1- Ajout d'indexes : provider et consumer

On commence par ajouter deux indexes sur les attributs **entryUUID** et **entryCSN**.

```
# cat index_replication.ldif
dn: olcDatabase={X}*db,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryUUID,entryCSN eq

# ldapmodify -Y EXTERNAL -H ldapi:/// -f  index_replication.ldif
```

ajuster {X}*db...

⇒ à faire sur chaque annuaire "provider (source) de réplication"

Petite vérification :

```
# ldapsearch -QLLL -b olcDatabase={2}mdb,cn=config -Y EXTERNAL -H ldapi:/// olcSuffix olcDbIndex
dn: olcDatabase={2}mdb,cn=config
olcSuffix: dc=example,dc=fr
olcDbIndex: entryUUID,entryCSN eq
olcDbIndex: objectClass eq,pres
olcDbIndex: ou,cn,mail,surname,givenname eq,pres,sub
```


Notes personnelles

9.2.2- Chargement du module "syncprov"

C'est lui qui fournit la fonctionnalité de maître de réplication :"sync provider"

```
# cat mod_syncprov.ldif
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: {0}syncprov.la

# ldapmodify -Y EXTERNAL -H ldapi:/// -f mod_syncprov.ldif
```

⇒ à faire sur chaque serveur maître

Petite vérification :

```
# ldapsearch -Q -Y EXTERNAL -H ldapi:/// -LLL -b cn=config "(olcModuleLoad=*)"
dn: cn=module{0},cn=config
objectClass: olcModuleList
cn: module{0}
olcModulePath: /usr/lib64/openldap
olcModuleLoad: {0}syncprov
```


Notes personnelles

9.2.3- Paramétrage de l'overlay sync provider pour l'annuaire source

On configure l'overlay pour chaque annuaire source de réplication (il a déjà été chargé par le serveur)

```
# cat syncprov-params.ldif
dn: olcOverlay=syncprov,olcDatabase={X}*db,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: {0}syncprov
olcSpCheckpoint: 100 10
olcSpSessionlog: 200

# ldapmodify  -Q -Y EXTERNAL -H ldapi:/// -f syncprov-params.ldif
```

⇒ à faire pour chaque annuaire "provider (source) de réplication"

syncprov-checkpoint 100 10 : si plus de 100 opérations / 10 minutes depuis le dernier checkpoint,
⇒ alors un checkpoint est provoqué.
Le checkpoint engendre d'écriture sur le disque de la valeur **contextCSN**.

syncprov-sessionlog 200 : le journal est dimensionné pour 200 entrées.

Notes personnelles

9.2.4- Création du compte de réplication

On crée un compte dédié[22] à la réplication à partir du fichier LDIF suivant (replicateur.ldif).

```
dn: ou=System,dc=example,dc=fr
ou: System
objectClass: organizationalUnit

dn: cn=replicator,ou=System,dc=example,dc=fr
cn: replicator
sn: REPLICATOR SN
objectClass: person
userPassword: {CRYPT}KsYn16wJg1fAY
```

⇒ à faire pour chaque annuaire "provider (source) de réplication"

Note : on choisit ici d'utiliser l'OU "System", dans laquelle se trouve ce compte.
De fait, pas de risque qu'il soit confondu avec un compte utilisateur (OU=People)

Ajouter cette entrée à l'annuaire :

```
# ldapadd -D cn=Manager,dc=example,dc=fr -w password -H ldapi:/// -f  replicator.ldif
```

Le mot de passe a été généré par la commande suivante :

```
# slappasswd -h {SSHA} -s "password"
```

22 C'est ce compte qui sera utilisé par le *consumer* (client) pour venir "aspirer" les données auprès du *provider.*

Notes personnelles

9.2.5- Mise en œuvre côté consumer

Pré-requis : L'annuaire cible de la réplication doit exister.
Les schémas utilisés dans le *provider* doivent être installés sur le *consumer.*

On déclare *le consumer* dans l'annuaire concerné :

"**olcSyncRepl:** …" dans l'entrée "dn: olcDatabase={X}*db,cn=config"

Ajouter l'attribut **olcSyncRepl** à l'annuaire cible :

```
dn: olcDatabase={X}*db,cn=config
changetype: modify
add: olcSyncRepl
OlcSyncrepl: rid=001
        provider="ldap://serveur-maitre"
        bindmethod=simple
        binddn="cn=replicator,ou=System,dc=example,dc=fr"
        credentials=password
        searchbase="ou=People,dc=example,dc=fr"
        filter="(objectClass=*)"
        scope=sub
        type=refreshOnly
        interval=00:00:10:00
        retry="60 10   300 3"
        schemachecking=off
```

Voir aussi : olcSyncUseSubentry, olcUpdateRef, et olcUpdateDN.

Notes personnelles

Paramètres :
Le **RID** : (Replication ID) un entier positif sur 3 digits.
Ce n'est qu'un numéro d'ordre de directive SyncRepl sur "ce *consumer* là".

Description de ce que l'on réplique :
La réplication porte sur tout objet, à partir de la base choisie (correspondant au filter)

Le **type** : 2 possibilités
refreshOnly : on planifie une synchro au bout d'un **intervalle** après la dernière réussie.
On spécifie donc un paramètre **interval** : 00:00:10:00 = 10 minutes
refreshAndPersist : on reste en flux tendu,
toute mise à jour du provider engendre sans délai une synchro du consumer.

Des paramètres de **retry** : des paires "temps" , "nombre d'essai".
"**60 10 300 3**" :
Toutes les **60** secondes pour **10** essais, puis toutes les **300** secondes pour **3** essais.
En cas de non-réussite au bout de ces 13 essais, on abandonne.

"**60 10 300 3 1800 +**" :
au bout des 13 échecs, on retente indéfiniment, mais toutes les demi-heures.

Note : Ne pas confondre **RID** (≤ 999) avec l'attribut **olcServerID** (SID) (≤ 4095). En mode multi-provider, chaque provider doit avoir un ServerID unique (visible dans le CSN, il indique d'où émane une modification, permet d'ignorer celles venant de nous-même, etc.).

Notes personnelles

9.3- Audit et Dépannage

9.3.1- Logs de réplication

Il est judicieux d'ajouter le niveau "**Sync**" à la liste des "*LogLevel*" suivis.

```
# cat logreplication.ldif
dn: cn=config
changetype: modify
add: olcLogLevel
olcLogLevel: Sync


# ldapmodify -Q -H ldapi:///   -Y EXTERNAL   -f logreplication.ldif
```

9.3.2- Quelques messages type côté Consumer

Problème de serveur non joignable (firewall) :

```
slap_client_connect: URI=ldap://192.168.157.94 DN="cn=replicator,ou=system,dc=example,dc=fr"
ldap_sasl_bind_s failed (-1)
```

Après ce type d'échec, nombre d'essais restant :

```
do_syncrepl: rid=001 rc -1 retrying (4 retries left)
```


Notes personnelles

Autre problème…

```
slap_client_connect: URI=ldap://u49-ldap-04.formation.actilis.fr
DN="cn=replicator,ou=system,dc=nodomain" ldap_sasl_bind_s failed (49)
```

⇒ Ne serait-ce pas le même message d'erreur sur le client lors d'un échec d'authentification?

Connexion réussie :

```
do_syncrep1: rid=001 starting refresh (sending cookie=rid=001)
```

Problème dans la requête de recherche :

```
do_syncrep2: rid=001 LDAP_RES_SEARCH_RESULT
do_syncrep2: rid=001 LDAP_RES_SEARCH_RESULT (32) No such object
do_syncrep2: rid=001 (32) No such object
```

⇒ Il se peut que le filtre ou la base de recherche soit erronée (côté Consumer)

Notes personnelles

Un refresh qui se passe bien :

```
syncrepl_message_to_entry: rid=001 DN: ou=People,dc=example,dc=fr, UUID: 48e28a04-4e82-103e-8fd2-
57eb50ee49e2
syncrepl_entry: rid=001 LDAP_RES_SEARCH_ENTRY(LDAP_SYNC_ADD) csn=(none) tid 0x7f21a51fc640
syncrepl_entry: rid=001 inserted UUID 48e28a04-4e82-103e-8fd2-57eb50ee49e2
syncrepl_entry: rid=001 be_search (32)
syncrepl_entry: rid=001 ou=People,dc=example,dc=fr
syncrepl_entry: rid=001 be_add ou=People,dc=example,dc=fr (32)
do_syncrep2: rid=001 LDAP_RES_SEARCH_RESULT
do_syncrep2: rid=001
cookie=rid=001,sid=001,csn=20240123202444.477384Z#000000#000#000000;20240123212950.658110Z#000000#00
1#000000
slap_queue_csn: queueing 0x7f2190109100 20240123212950.658110Z#000000#001#000000
slap_graduate_commit_csn: removing 0x7f2190109100 20240123212950.658110Z#000000#001#000000
```

Si on a 2 RID définis sur le consumer, on voit que la synchro alterne de l'un à l'autre (rescheduling…)

Quand l'un est en cours, l'autre est mis en pause :

```
start_refresh: rid=002 a refresh on rid=001 in progress, pausing
```

⇒ le 002 est mis en pause tant que le 001 est en cours.

Ici, le 001 a fini, le 002 peut démarrer :

```
refresh_finished: rid=001 rescheduling refresh on rid=002
```


Notes personnelles

9.3.3- Messages côté Provider

Un consumer s'est connecté ! Et on avait des données à lui transmettre :

```
conn=1027 op=1 syncprov_op_search: got a search with a cookie=rid=001
conn=1027 op=1 syncprov_search_response:
cookie=rid=001,sid=001,csn=20240123202444.477384Z#000000#000#000000;20240123212950.658110Z#000000#00
1#000000
```

9.3.4- Statut de la réplication

On peut s'appuyer sur le contextcsn de part et d'autre.

```
$ ldapsearch -x -LLL -b dc=example,dc=fr -s base -D cn=Manager,dc=example,dc=fr -w password -H
ldap://192.168.157.94 contextcsn
dn: dc=example,dc=fr
contextCSN: 20240123202444.477384Z#000000#000#000000
contextCSN: 20240123213741.528350Z#000000#001#000000

$ ldapsearch -x -LLL -b dc=example,dc=fr -s base -D cn=Manager,dc=example,dc=fr -w password -H
ldap://192.168.157.95 contextcsn
dn: dc=example,dc=fr
contextCSN: 20240123202444.477384Z#000000#000#000000
contextCSN: 20240123213741.528350Z#000000#001#000000
```

Ici, ils annoncent la même chose, la réplication est synchrone.

Notes personnelles

9.3.5- Quand on est synchronisé

Dans les logs côté *provider*

```
conn=1134 op=1 syncprov_op_search: got a search with a
cookie=rid=001,csn=20240123202444.477384Z#000000#000#000000;20240123213741.528350Z#000000#001#000000
conn=1135 op=1 syncprov_op_search: got a search with a
cookie=rid=002,csn=20240123202444.477384Z#000000#000#000000;20240123213741.528350Z#000000#001#000000
```

Dans les logs côté *consumer*

```
do_syncrep1: rid=001 starting refresh (sending
cookie=rid=001,csn=20240123202444.477384Z#000000#000#000000;20240123213741.528350Z#000000#001#000000
)
do_syncrep2: rid=001 LDAP_RES_SEARCH_RESULT
do_syncrep1: rid=002 starting refresh (sending
cookie=rid=002,csn=20240123202444.477384Z#000000#000#000000;20240123213741.528350Z#000000#001#000000
)
do_syncrep2: rid=002 LDAP_RES_SEARCH_RESULT
```


Notes personnelles

C'est une bonne idée d'avoir renseigné des "olcServerID" sur nos serveurs, on les voit dans les CSN figurant dans les logs.

Exemple sur un slapd 2.6, avec un syncrepl déclaré en refreshAndPersist :

Provider :

```
Feb 07 14:30:17 u49-ldap-04.formation.actilis.fr slapd[23007]: conn=1097 op=1 syncprov_qresp: set up
a new syncres mode=4 csn=20240207143017.758054Z#000000#004#000000
Feb 07 14:30:17 u49-ldap-04.formation.actilis.fr slapd[23007]: conn=1097 op=1 syncprov_sendinfo:
sending a new
cookie=rid=001,sid=004,csn=20240207142531.862163Z#000000#000#000000;20240207143017.758054Z#000000#00
4#000000
```

Consumer :

```
do_syncrep2: rid=001 LDAP_RES_INTERMEDIATE - NEW_COOKIE
Feb 07 14:30:17 u49-ldap-05.formation.actilis.fr slapd[1964]: do_syncrep2: rid=001 NEW_COOKIE:
rid=001,sid=004,csn=20240207142531.862163Z#000000#000#000000;20240207143017.758054Z#000000#004#00000
0
```


Notes personnelles

9.3.6- Mise à jour côté Consumer

Modifier une base consumer…n'est pas souhaitable !

En cas de requête de modification sur les clients de réplication, on peut renvoyer un **Referral** vers le maître. L'attribut **olcUpdateRef** permet de spécifier une liste d'urls de renvoi vers un maître..

Relancer la réplication en cas de désynchro

On peut relancer une réplication complète en spécifiant le cookie au démarrage de slapd. Il fait porter attention à l'identité, et pour cela, on peut reprendre la ligne de commande du service :

```
# systemctl stop slapd
# systemctl cat slapd.service | grep ^ExecStart=
ExecStart=/usr/sbin/slapd -u ldap -h "ldap:/// ldaps:/// ldapi:///"
# /usr/sbin/slapd -u ldap -h "ldap:/// ldaps:/// ldapi:///" -c rid=001,csn=0
```

Pour arrêter :

```
# pkill slapd
# rm /run/ldapi[23]
```

La même chose ensuite pour les autres "rid" sur lesquels on est consumer.

23 Sous CentOS/RedHat/Alma, si SELinux est activé, le champ type du contexte de sécurité (ls -Z) de /run/ldapi créé en démarrant le serveur "manuellement" est "var_run_t", et non pas "slapd_var_run_t". Cela empêche un démarrage du service par *systemctl*. D'où la suppression de /run/ldapi pour permettre un redémarrage "normal".

Notes personnelles

10- Utilisation de LDAP dans les applications

Notes personnelles

10.1- Authentification Apache basée sur LDAP

Cette fonctionnalité est fournie en standard la distribution d'Apache HTTP Server.

Le module **mod_ldap** apporte la gestion des connexions et le caching des résultats. Les fonctionnalités liées à l'authentification sont proposées par le module **mod_authnz_ldap**.

Installation :
Dans le monde Debian, ces modules sont fournis par le package **apache2**.
Dans le monde RedHat, ils ne sont pas inclus dans le package **httpd**, mais fournis par **mod_ldap**.

En cas de compilation :
Les deux modules doivent donc être choisis lors du « configure » de la compilation :

```
--with-ldap
--enable-ldap
--enable-authnz-ldap
```

Le "**--with-ldap**" est important, sinon la compilation échouera[24].

Sécurité / firewalls
Le module suppose que l'on dispose d'un annuaire LDAP accessible depuis le serveur HTTP.

24 Pensez aux packages fournissant les includes de développement OpenLDAP (openldap-devel), et si vous rencontrez des problèmes, indiquez l'option "--with-ldap" avant le premier "enable". Forcez ensuite une compilation complète par make clean ; make".

Notes personnelles

10.1.1- Fonctionnement de mod_authnz_ldap

La vérification d'une identité via LDAP fonctionne en deux phases :

10.1.1.1- La phase d'authentification : search() + bind()

Le module **recherche** dans l'annuaire une entrée correspondant au nom d'utilisateur saisi
Si aucune ou plus d'une entrée est trouvée, l'accès est interdit
Sinon, le module tente un **ldapbind**() en utilisant ce nom et le mot de passe fourni
Si le ldapbind() échoue, l'accès est interdit

10.1.1.2- La phase d'autorisation : ldapcompare()

Le module réalise des **opérations de comparaison** pour déterminer si le compte sous lequel le ldapbind() a eu lieu dispose des privilèges requis par la directive **Require**.

10.1.2- Configuration de mod_authnz_ldap

10.1.2.1- Déclarer l'authentification

On place dans le contexte où l'on souhaite déclencher l'authentification un bloc classique avec :
Le royaume : **AuthName** "Veuillez vous identifier dans l'annuaire" (par exemple)
Le type d'authentification : **AuthType** Basic (par exemple)
Les contrôles requis : **Require** valid-user (par exemple)
Sans oublier, depuis Apache 2.2 : **AuthBasicProvider ldap**
Qui définit que c'est par LDAP que nous vérifions les autorisations.

Notes personnelles

10.1.2.2- Paramétrer la phase d'authentification

La directive **AuthLDAPUrl**[25] est utilisée pour générer la requête de recherche.

Elle attend une URL LDAP tel que définies par la RFC 2255 :

```
AuthLDAPUrl ldap[s]://host:port/basedn?attribute?scope?filter [NONE|SSL|TLS|STARTTLS]
```

Écrire "**ldap://**… … … **SSL**" est équivalent à écrire "**ldaps://**... … …"

L'URL doit comporter : le nom **d'hôte**, le **port**, la **base**, l'**attribut recherché** et les **filtres**

La requête (search) de l'*attribut* **doit obtenir une unique réponse**, sinon, l'accès est refusé.

Le champ retourné est le DN utilisé pour l'étape suivante : on tente ensuite de se connecter (bind) à l'annuaire avec ce DN et le mot de passe fourni par le client. Si cela échoue, l'accès est refusé.

Identité pour la recherche dans l'annuaire

La phase de recherche est réalisée de manière anonyme (sauf si l'URL LDAP précise un DN)

On peut choisir un DN de connexion à l'annuaire pour réaliser la recherche grâce aux directives **AuthLDAPBindDN** et **AuthLDAPBindPassword**.

25 Voir http://httpd.apache.org/docs/2.2/mod/mod_authnz_ldap.html#authldapurl

Notes personnelles

10.1.2.3- Paramétrer la phase d'autorisation (la directive Require)

Cette phase de comparaison dépend de ce que l'on spécifie dans la directive « Require » :

Les 3 premiers cas ne nécessitent pas de comparaison auprès de l'annuaire :
require valid-user : directive par défaut, ne nécessite aucune opération de comparaison

require user xxxx : le nom d'utilisateur "xxxx" doit être identique à celui "fetché". L'UID recherché dans l'annuaire doit faire partie de ceux listés.

```
require user fmicaux mgarnier
```

require dn : le DN spécifié doit être celui "fetché" dans l'annuaire

```
require dn cn=fmicaux,ou=users,dc=example,dc=fr
```

Ces deux derniers cas nécessitent une comparaison :
require group1 : le DN "fetché" doit appartenir au groupe "group1" (dans l'annuaire)
Voir les directives **AuthLDAPGroupAttribute** et **AuthLDAPGroupAttributeIsDN**.

require ldap-attribute : la comparaison de l'attribut (du DN fetché) doit réussir.

```
require ldap-attribute employeeType=active
```


Notes personnelles

10.1.2.4- Directives du module mod_authnz_ldap (valeurs par défaut en gras)

AuthLDAPGroupAttribute : Attribut LDAP utilisé pour vérifier l'appartenance à un groupe. Si rien n'est spécifié, les attributs **member** ou **uniquemember** sont utilisés.

AuthLDAPGroupAttributeIsDN **On**/Off : utiliser le DN du client (au lieu du nom d'utilisateur fourni) pour rechercher son appartenance à un groupe.

AuthLDAPCompareDNOnServer **On**/Off : lors de la phase d'autorisation, la comparaison des DN a lieu entre le DN résultant de l'authentification et celui trouvé par la recherche liée au critère "require DN" et elle est faite par le serveur LDAP. Si Off, c'est une comparaison de chaîne de caractères effectuée par le module entre le DN résultant de l'authentification et celui retourné par la recherche.

AuthLDAPRemoteUserIsDN On/**Off** : utiliser le DN de l'utilisateur authentifié pour alimenter la variable REMOTE_USER, au lieu du nom d'utilisateur fourni par l'utilisateur.

AuthLDAPDereferenceAliases never, searching, finding, **always**
AuthLDAPCharsetConfig : Fichier de configuration concernant les conversions de caractères

Notes personnelles

10.2- Authentification Squid basée sur LDAP

La déclaration d'une authentification dans Squid se fait par le biais d'une "**acl**" à laquelle on associe une décision par "**http_access**"...

```
acl Users     proxy_auth REQUIRED
http_access   allow Users
```

"**proxy_auth**" est un type d'ACL paramétrable par la directive "**auth_param**". On doit notamment déclarer quelques éléments sur la durée de vie d'une session authentifiée...

```
auth_param basic credentialsttl 2 hours
auth_param basic casesensitive on
auth_param basic children 5
auth_param basic realm AUTHENTIFICATION PROXY
```

La directive **auth_param *program*** présente **un exécutable** dont le rôle est d'attendre (stdin) des couples login+password, de réaliser le contrôle, et d'annoncer (stdout) un verdict par OK ou ERR.

Concernant l'authentification LDAP, Squid est fourni avec un "program" **squid_ldap_auth** et un "program" **squid_ldap_group** que l'on trouve dans le répertoire **/usr/lib/squid**.

```
squid_ldap_auth -b "base DN" [-u attribute] [options] [ldap_server_name[:port]|URI]...
```

Exemple :

```
auth_param basic program /usr/lib/squid/squid_ldap_auth -b "dc=dom0,dc=fr" -f "uid=%s"
```


Notes personnelles

11- Programmer avec LDAP

Notes personnelles

11.1- Utiliser LDAP en C

Cet exemple de code illustre un parcours d'annuaire à partir d'une branche (ou=People, dc=example, dc=fr).

Opérations réalisées :

- On se connecte à l'annuaire,
- On parcourt toutes les entrées
- Pour chacune on liste tous les attributs
- On se déconnecte et on libère les ressources

Pour le compiler, il faut le package openldap-devel (RedHat) ou libldap2-dev (Ubuntu)

Puis compiler comme ceci :

```
$ gcc -lldap -llber ldap.c -o exemple-ldap
```

Exécution :

```
$ ./exemple-ldap
```


Notes personnelles

```c
#include <stdio.h>
#include <ldap.h>

const char *LDAP_URI = "ldap://192.168.0.100";
const char *BIND_DN = "cn=ldapadmin,dc=example,dc=fr" ;
const char *BIND_PW = "secret";
const char *SEARCH_BASE= "dc=example,dc=fr" ;

const char *attrs[] = { "*", "+", NULL };

int
main ()
{
    LDAP *conn;
    LDAPMessage *res;
    LDAPMessage *entry;
    int ret;

    /* Connexion au server */
    ldap_initialize (&conn, LDAP_URI);

    /* Bind */
    ldap_simple_bind_s (conn, BIND_DN, BIND_PW);

    /* Initialisation de la recherche */
    ret = ldap_search_s (conn, SEARCH_BASE,
                   LDAP_SCOPE_ONELEVEL, NULL,
                   attrs, 0, &res);
    if (ret != LDAP_SUCCESS)
    {
        printf ("Search failed\n");
        ldap_unbind (conn);
        conn = NULL;

        return 1;
    }

    /* Itération sur chacune des entrées du résultat de la recherche */
    for (entry = ldap_first_entry (conn, res);
         entry;
         entry = ldap_next_entry (conn, entry))
    {
        BerElement *ber;
        char *attr;
        char *dn = NULL;

        dn = ldap_get_dn (conn, entry);
        printf ("dn: %s\n", dn);
        ldap_memfree (dn);

        /* Itération sur chaque attribut de l'entrée */
        for (attr = ldap_first_attribute (conn, entry, &ber);
             attr;
             attr = ldap_next_attribute (conn, entry, ber))
        {
            /* La structure berval permet de manipuler tout type de données (y
             * compris binaires) */
            struct berval **values;
            struct berval **p;

            values = ldap_get_values_len (conn, entry, attr);
            for (p=values; *p; p++)
                printf ("%s: %s\n", attr, (*p)->bv_val);
            ldap_value_free_len (values);
        }

        ldap_memfree (attr);
        ber_free (ber, 0);

        printf ("\n");
    }

    /* Libération des ressources */
    ldap_msgfree (res);
    ldap_unbind (conn);
    conn = NULL;

    return 0;
}
```

11.2- Utiliser LDAP en PHP

11.2.1- Se connecter à l'annuaire et s'authentifier

La fonction **ldap_connect** se connecte puis la fonction **ldap_bind** s'authentifie : bind.php

```
<?php
// Paramètres
$ldaprdn = 'cn=admin,dc=example,dc=fr';
$ldappass = 'password';
$ldapserver = 'localhost';

// Connection
$ldapconn = ldap_connect($ldapserver);

// retourne un identifiant de connexion si ça s'est bien passé.
if (! $ldapconn) {
  echo "Connect failed...\n";
} else {
  // Passage en LDAP V3
  ldap_set_option($ldapconn, LDAP_OPT_PROTOCOL_VERSION, 3);
  $ldaprc = ldap_bind($ldapconn, $ldaprdn, $ldappass);
  // CR=0 si OK, autre chose sinon
  if ($ldaprc) {
    echo "LDAP bind successful...\n";
  } else {
    echo "LDAP bind failed...\n";
  }
}
?>
```


Notes personnelles

11.2.2- Se connecter en anonyme

Au lieu de :

```
$ldaprc = ldap_bind($ldapconn, $ldaprdn, $ldappass);
```

utiliser :

```
$ldaprc = ldap_bind($ldapconn);
```

11.2.3- Se déconnecter ldap_unbind()

Attention, **ldap_unbind** détruit le descripteur de connexion.
Après ça, il faut recréer la connexion si on en a encore besoin... (ldap_connect).

unbind.php :

```
// On peut maintenant se déconnecter, sauf si on a autre chose à réaliser
$ldaprc = ldap_unbind ( $ldapconn );
```

Si il s'agit d'un simple changement d'utilisateur, on se "re"-bind, on ne fait pas de unbind

La fonction **ldap_close()** est un alias pour **ldap_unbind()**.

Notes personnelles

11.2.4- Les fonctions ldap_search() et ldap_get_entries()

Elle permettent d'effectuer le même type de recherche que la commande **ldapsearch**.

Exemple

```
<?php
  // On se connecte
  include "bind.php";

  $base="dc=example,dc=fr";
  $filter="objectClass=*";
  $attributes = array("dn");
  $search_result=ldap_search($ldapconn, $base, $filter, $attributes);

  // Traitement des résultats
  $info = ldap_get_entries($ldapconn, $search_result);
  echo $info["count"]." entries returned\n";

  // Listage des données
  for ($i = 0; $i<$info["count"]; $i++) {
    echo $info[$i]["dn"]."\n";
  }

  include "unbind.php";
?>
```

Il existe d'autres fonctions pour lister les valeurs :

ldap_get_dn : donne directement le DN d'une entrée

ldap_first_entry & **ldap_next_entry** : qui retournent un entryID

On les associe à **ldap_get_values**, pour faire le même traitement que dans l'exemple.

Notes personnelles

12- Administration de comptes utilisateurs

Notes personnelles

12.1- LDAP Account Manager

12.1.1- Présentation

LAM (http://www.ldap-account-manager.org/) est une application web d'administration graphique pour les comptes basés dans un annuaire LDAP.

On peut l'utiliser comme navigateur LDAP, et elle offre en standard des formulaires pour :

- les utilisateurs et groupes.
- les domaines et les machines
- des extensions Asterisk et Messagerie électronique sont également disponibles

L'application permet de manipuler des comptes systèmes Unix, Samba, DHCP, gère les clés SSH, et permet de gérer les comptes des applications groupware comme phpgroupware et Kolab, ou encore les comptes DHCP.

Des packages[26] sont disponibles pour Debian, Fedora ou SuSE.

Une image Docker est aussi disponible et permet de spécifier l'essentiel de la configuration grâce à des variables d'environnement. Voir : https://hub.docker.com/r/ldapaccountmanager/lam.

Pré-requis: disposer de Docker sur la machine cible.

26 L'installation de la version packagée ou depuis le source du projet (https://github.com/LDAPAccountManager/) nécessite un serveur HTTP, et PHP, mais pas forcément Docker, utilisé ici pour simplifier le déploiement.

Notes personnelles

12.2- Déploiement de LAM

12.2.1- Installer Docker

```
# curl -sSL https://get.docker.com | sh
```

12.2.2- Préparer le déploiement

Créer un dossier "lam", dans lequel on crée un fichier nommé "**compose.yaml**" avec le contenu donné page suivante.

12.2.3- Déployer

```
# docker compose up -d
```

12.2.4- Firewall

S'assurer que le firewall (s'il est actif) autorise à contacter le service http :

```
# firewall-cmd --add-service=http --premanent
# firewall-cmd --reload
```


Notes personnelles

Le fichier "compose.yaml" :

```
version: '3.9'

volumes:
  config:

services:
  webapp:
    image: ldapaccountmanager/lam:stable
    environment:
    - LAM_SKIP_PRECONFIGURE=false
    - LDAP_DOMAIN=example.fr
    - LDAP_BASE_DN=dc=example,dc=fr
    - LDAP_USERS_DN=ou=People,dc=example,dc=fr
    - LDAP_GROUPS_DN=ou=Group,dc=example,dc=fr
    - LDAP_SERVER=ldaps://ldap.example.fr:636
    - LDAP_USER=cn=manager,dc=example,dc=fr
    - LAM_LANG=fr_FR
    - LAM_CONFIGURATION_DATABASE=files
    - LAM_DISABLE_TLS_CHECK=false
    - LDAP_ORGANISATION=LDAP Account Manager Demo
    - LAM_PASSWORD=lam
    - LDAP_ADMIN_PASSWORD=123Soleil
    - LDAP_READONLY_USER_PASSWORD=readonlypw

  volumes:
  - type: volume
    source: config
    target: /var/lib/ldap-account-manager/config

  network_mode: host
```


Notes personnelles

13- Annexes

Notes personnelles

13.1- Le format slapd.conf

Le format **slapd.conf** est documenté par la page de manuel **slapd.conf (5)**.

Il peut être divisé en plusieurs sections

- Au début du fichier, une section de type « configuration globale »
- Les inclusions de schémas
- Ensuite, une section concernant les **backends**
- Puis une section par base de données gérées (**database**)
  - Chacune se termine au début de la suivante (ou à la fin du fichier)
  - Chacune définit un annuaire (avec ses fichiers, sa racine, ses permissions)
  - Chacune définit ses permissions (acl)

13.1.1- Les schémas

Introduits par la directive « **include** », des sous-fichiers de configuration peuvent être ajoutés à la configuration. C'est le cas pour les fichiers qui définissent le schéma (type de données gérées)

Plusieurs schémas sont fournis en standard (voir **/etc/openldap/schema**)

```
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/misc.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/nis.schema
schemacheck  on
```


Notes personnelles

13.1.2- Les logs

Ils sont envoyés à **syslogd**, sous la facilité « **LOCAL4** ». Voir **/etc/rsyslog.conf**.

```
local4.*                                        /var/log/ldap.log
```

La directive **LogLevel** spécifie le niveau de log souhaité façon puissances de 2. Pour demander des traces sur la gestion des ACLS et sur le fichier de configuration : 128 + 64 = 192

Valeur	Fonction correspondante
1	Appels de fonctions
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

Notes personnelles

13.1.3- Backends

Il s'agit du format de stockage des bases de données de l'annuaire.

BDB, a succédé à LDBM (plus fourni) et a lui même été remplacé par HDB.
HDB utilise une structure hiérarchique et permet de renommer des branches de l'arborescence.
MDB (bibliothèque Lightning Memory-Mapped DB), est aujourd'hui par défaut.

13.1.4- Les sections database

Chaque section database définit un annuaire que slapd peut gérer. La fin d'une section database est atteinte au prochain mot « database » ou à la fin de **slapd.conf**

```
database        hdb
directory       /var/lib/ldap
```

13.1.5- Droits des répertoires de stockage

Le répertoire cité par « directory » doit exister avant le démarrage de slapd.
Il devrait être en mode « 700 » (et propriété du user sous lequel tourne slapd)

```
# ls -ld /var/lib/ldap/
drwx------  2 ldap ldap 4096 Mar 17 23:08 /var/lib/ldap/
```

Extrait d'un « ps faux »

```
ldap 31930 0.0 1.5 13416 3928 ? Ssl 22:05 0:00 /usr/sbin/slapd -u ldap -h ldap:///
```


Notes personnelles

13.1.6- La racine d'une database

Spécifiée par la directive « suffix », elle correspond en général au nom de domaine Internet.

```
suffix          "dc=actilis,dc=net"
```

13.1.7- L'administrateur d'une database

Il s'agit du super-utilisateur (équivalent de **root** sous Unix) de l'annuaire.

Il n'est **pas soumis aux ACLs**
On le déclare par les directives **rootdn** et **rootpw**
Il est souvent virtuel (pas d'entrée portant son DN dans l'annuaire)

```
rootdn          "cn=ldapadmin,dc=actilis,dc=net"
```

Le mot de passe peut être stocké en clair ...

```
rootpw          secret
```

ou sous forme hachée (forme générée par **slappasswd**)

```
[root@vm1-centos4 openldap]# slappasswd -s "password" -h {CRYPT}
{CRYPT}ONHMD5XB/E8vk
```

Et donc dans le fichier slapd.conf :

```
rootpw          {CRYPT}ONHMD5XB/E8vk
```


Notes personnelles

13.1.8- Indexation des bases

Les index permettent d'**accélérer les recherches** dans l'annuaire

La directive **index** introduit le nom d'un attribut et un critère qui sera souvent utilisé

```
index           objectClass eq
```

On peut préciser un critère pour plusieurs atributs, dans ce cas, on factorise le critère :

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

Types d'index :

Index	Type de recherche (filtre)
eq	égalité stricte ('uid=fmicaux')
sub	utilisation d'un wildcard ('uid=f*')
subinitial	optimisation de sub pour wildcard à la fin ('uid=fmi*')
subfinal	optimisation de sub pour wildcard au début ('uid=*aux')
subany	optimisation de sub pour wilcard au début ou à la fin ('uid=*ic*')
approx	recherche par approximation sonore ('uid~=miko')
pres	recherche de présence ('objectclass=posixAccount')

Notes personnelles

13.1.9- Contrôle d'accès aux bases

Les **ACLs** permettent de **définir des droits d'accès à l'annuaire**. Elles font l'objet d'une page de manuel : **slapd.access(5)**.

```
access to <what> [ by <who> <access> [ <control> ] ]+
```

L'ordre est très important : Elles sont évaluées dans leur ordre de déclaration.
Dès qu'une ACL correspond à la cible recherchée (**to**), la recherche est arrêtée.
On déclare donc les ACLS générales **APRÈS** les ACLS précises.

Exemple : On autorise l'accès à l'attribut « *userPassword* »
pour l'authentification (**auth**) des utilisateurs non encore authentifiés (anonymous)
on accorde le droit de le modifier (**write**) son propre mot de passe (self)
aucun accès (**none**) pour les autres utilisateurs (*****)

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

Exemple : On accorde à tout le monde (by *****) le droit de lire tout l'annuaire (to *****)

```
access to *
            by * read
```


Notes personnelles

13.2- Oublier le passé et migrer vers olc

13.2.1- Migrer de slapd.conf vers slapd.d

Extrait du man de **slapd** (et de **slaptest**)

```
-f slapd.conf
       specify an alternative slapd.conf(5) file.

-F confdir
       specify a config directory.  If both -f and -F  are  speci-
       fied,  the config file will be read and converted to config
       directory format and written to  the  specified  directory.
       If  neither  option  is specified, slaptest will attempt to
       read the default config directory before trying to use  the
       default  config  file.  If  a valid config directory exists
       then the default config file is ignored. If dry-run mode is
       also specified, no conversion will occur.
```

Avant la version 2.4, Slapd démarrait avec l'option "-f" pointant un fichier de configuration (**slapd.conf**).

Ce fichier n'est plus fourni par les packages.

Dans les versions >= 2.4, la configuration est basée sur un répertoire de configuration ('**slapd.d**') contenant des fichiers au format LDIF. On parle de "OLC" (Online Configuration).

On peut encore utiliser[27] l'ancien format comme point de départ, peut-être plus lisible, mais l'usage veut qu'on convertisse les configurations existantes vers le nouveau format, dynamique.

27 À condition de disposer / créer un fichier slapd.conf

Notes personnelles

13.2.2- Une approche "slapd.conf" pour (re)commencer

1/ Supprimer le répertoire /etc/openldap/slapd.d (ou /etc/ldap/slapd.d) et le répertoire /var/lib/ldap.

```
# systemctl stop slapd
# rm -rf  /etc/*/slapd.d /var/lib/ldap
```

2/ Créer les nouveaux répertoires :

```
# install -d -m 700 -o ldap -g ldap  /etc/[open]ldap/slapd.d /var/lib/ldap/{dom1.fr,dom2.fr}
```

3/ Créer un fichier (/root/init.conf par exemple) avec le contenu suivant :

```
database config
rootdn "cn=admin,cn=config"
rootpw configpass
pidfile /var/run/openldap/slapd.pid
loglevel 768

include  /etc/ldap/schema/core.schema
include  /etc/ldap/schema/cosine.schema
include  /etc/ldap/schema/inetorgperson.schema
include  /etc/ldap/schema/nis.schema

# 1er annuaire
database        hdb
suffix          "dc=dom1,dc=fr"
rootdn          "cn=admin,dc=dom1,dc=fr"
rootpw          passdom1
directory       "/var/lib/ldap/dom1.fr"
dbconfig set_cachesize 0 2097152 0
index           objectClass eq
```


Notes personnelles

```
# 2nd annuaire
database        hdb
suffix          "dc=dom2,dc=fr"
rootdn          "cn=admin,dc=dom2,dc=fr"
rootpw          passdom2
directory       "/var/lib/ldap/dom2.fr"
dbconfig set_cachesize 0 2097152 0
index           objectClass eq
```

4/ Exécuter les suivantes : (conversion vers cn=config, création des bases, modifs permissions)

```
# slaptest  -n 0 -f init.conf -F /etc/ldap/slapd.d
config file testing succeeded

# chown -R ldap.ldap /var/lib/ldap /etc/ldap/slapd.d
# chmod 750          /var/lib/ldap /etc/ldap/slapd.d
```

5/ Démarrer Slapd

```
# systemctl start slapd
```

On dispose en quelques secondes de deux annuaires hébergés sur le même serveur.

```
# ldapwhoami -x -D "cn=admin,dc=dom1,dc=fr" -w passdom1  -h localhost
dn:cn=admin,dc=dom1,dc=fr
# ldapwhoami -x -D "cn=admin,dc=dom2,dc=fr" -w passdom2  -h localhost
dn:cn=admin,dc=dom2,dc=fr
```

6/ **Administrer la suite dynamiquement…**

Notes personnelles

Index lexical

Notes personnelles