

Delphine FONTAINE
GM3



Génération d'un annuaire (ldap)



32 rue Raymond Aron
Parc d'activité Technologiques de la Vatine
76130 MONT SAINT AIGNAN



Stage technicien
Département Génie Mathématiques, 3^{ème} année
20 juillet-20 septembre 1998

REMERCIEMENTS :

Je voudrais remercier Monsieur Dany Vandromme pour m'avoir accueillie au sein de son équipe.

Je tiens à remercier également toute l'équipe du CRIHAN qui m'a permis d'effectuer mon stage dans une ambiance de travail agréable.

Je remercie Philippe Wender et Philippe Savary pour leur aide sur la partie INSA de mon travail.

Enfin, je remercie plus particulièrement Madame Béatrice Charton pour son soutien et pour avoir assuré le succès de ce stage.

SOMMAIRE

REMERCIEMENTS :	1
INTRODUCTION :	5
CHAPITRE I : LE CRIHAN	6
I) LE LIEU :	6
II) LE RÔLE :	7
III) L'ÉQUIPE :	8
CHAPITRE II : LDAP (LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL)	9
I) PRÉSENTATION DE LDAP :	9
II) COMMENT LES SERVEURS LDAP ORGANISENT-ILS LES ANNUAIRES ?	10
III) SLAPD, SLURPD ET X.500.....	12
IV) LES DIFFÉRENTES CONFIGURATIONS POSSIBLES :	13
V) LES DIFFÉRENTS CLIENTS :	14
CHAPITRE III : L'INSTALLATION	16
I) <i>PREMIÈRE ÉTAPE</i> : RAMENER LE PACKAGE LDAP.....	16
II) <i>DEUXIÈME ÉTAPE</i> : LIRE LES FICHIERS README ET INSTALL	17
III) <i>TROISIÈME ÉTAPE</i> : MODIFIER LES FICHIERS DE CONFIGURATION.	18
IV) <i>QUATRIÈME ÉTAPE</i> : TÉLÉCHARGEMENT DE GDBM-1.7.3.	18
V) <i>CINQUIÈME ÉTAPE</i> : INSTALLER GDBM.	19
VI) <i>SIXIÈME ÉTAPE</i> : MAKE.....	21
VII) <i>SEPTIÈME ÉTAPE</i> : MAKE TEST.	24
VIII) <i>HUITIÈME ÉTAPE</i> : MAKE INSTALL.	24
CHAPITRE IV : LE FICHIER MAKE-COMMON	26
I) LES CHEMINS D'INSTALLATION DE LDAP :	26
II) LE PACKAGE KERBEROS :	26
III) LE PACKAGE ISODE :	26
IV) LA PARTIE CONCERNANT SLAPD :	27
V) LE CLIENT XAX500 :	28
VI) LES CARACTÈRES :	28
VII) LES OPTIONS :	28
CHAPITRE V : CONFIGURATION UTILISÉE ET PROBLÈMES RENCONTRES	29
I) MATÉRIEL UTILISÉ :	29
II) CONFIGURATION :	29
III) PROBLÈMES RENCONTRES :	32
IV) LA LISTE LDAP.....	37
CHAPITRE VI : LE SERVEUR SLAPD	38
I) DÉMARRER ET ARRÊTER LE SERVEUR :	38
II) PROBLÈMES RENCONTRÉS LORS DU DÉMARRAGE DU SERVEUR :	41
III) FICHIER DE CONFIGURATION :	42
CHAPITRE VII : CRÉATION D'UNE BASE DE DONNÉES	45
I) LE FORMAT LDIF :	45
II) CRÉATION D'UNE BASE DE DONNÉES AVEC LA COMMANDE LDIF2LDBM :	49
III) UTILISATION DE LA COMMANDE LDBMCAT :	50
CHAPITRE VIII : LES COMMANDES LDAP	51
I) UTILISATION DU FORMAT LDIF :	51

II) LA COMMANDE LDAPSEARCH :	51
III) LA COMMANDE LDAPADD :	55
IV) LA COMMANDE LDAPDELETE :	57
V) LA COMMANDE LDAPMODRDN :	58
VI) COMMANDE LDAPMODIFY :	59
CHAPITRE IX : LES ACCENTS	65
I) DÉCOUVERTE DU PROBLÈME :	65
II) DEMANDE D'AIDE À LA LISTE :	66
III) RÉCUPÉRATION DES FICHIERS NÉCESSAIRES :	67
IV) LE FORMAT UTF-8 :	68
V) NOUVEAU PROBLÈME :	68
VI) CHOIX D'UN COMPROMIS :	70
CHAPITRE X : LE SERVEUR WEB	71
I) INSTALLATION DU SERVEUR :	71
II) EMBLEMMENT DES FICHIERS :	71
III) LA PAGE D'ACCUEIL DE L'ANNUAIRE :	72
IV) LA RECHERCHE D'UNE PERSONNE :	73
V) L'AJOUT D'UNE PERSONNE DE L'INSA :	75
VI) LA MODIFICATION D'UNE PERSONNE :	78
a) <i>Ajouter un champ</i> :	78
b) <i>Changement et suppression d'un attribut</i> :	81
VII) SUPPRESSION D'UNE PERSONNE :	82
VIII) LE LIEN "VERS L'ANCIEN ANNUAIRE" :	86
CHAPITRE XI : QUELQUES SCRIPTS UTILES	87
I) CONVERS.PL :	87
II) LIEU.PL :	89
III) TRI.PL :	89
IV) LDBM2LDIF:	90
V) GENERE_LDBM:	91
CONCLUSION	92
GRILLE DE DEROULEMENT DU STAGE	93
ANNEXE	ERROR! BOOKMARK NOT DEFINED.

INTRODUCTION :

Ce stage technicien s'est déroulé du 20 juillet au 20 septembre 1998 au sein du CRIHAN (Centre de Ressources Informatiques de Haute Normandie).

Il s'agissait de **générer un annuaire électronique, accessible en réseau, basé sur le protocole LDAP**. Ce sont donc essentiellement les acquis informatiques qui ont été exploités et développés lors de ce stage.

Ce type d'annuaire contient des informations sur les personnes mais peut aussi en contenir sur les ressources, telles que les salles de réunion ou le matériel informatique. Le CRIHAN souhaitait y trouver essentiellement les numéros de téléphone, les adresses électroniques etc. ...

LDAP (Lightweight Directory Access Protocol) est un protocole de gestion d'annuaires développé par l'Université du Michigan. Il permet également aux "clients" LDAP d'effectuer des recherches sur le serveur dans des conditions particulièrement performantes. Outre le client LDAP proprement dit, qui n'est disponible que sur station de travail, les outils de messagerie électronique les plus connus, comme ceux de Netscape, Microsoft Internet Explorer ou Eudora, intègrent aujourd'hui des clients LDAP.

Mon travail comportait plusieurs parties :

- L'installation de la partie serveur de LDAP sur une machine Unix.
- Les tests de base sur un annuaire fictif me permettant de vérifier les fonctionnalités des quelques clients LDAP mentionnés ci-dessus pour chaque plate-forme. Ceux-ci m'ont permis de me familiariser avec ce protocole.
- Le rendu opérationnel d'un serveur pour le CRIHAN.

Les performances de la première maquette ayant aussitôt attiré l'attention de Philippe Wender, responsable informatique et réseaux de l'INSA de Rouen, celui-ci m'a demandé d'en développer une version pour cette école. L'INSA possédait déjà un annuaire sur l'Intranet, mais uniquement constitué de pages HTML dont la mise à jour nécessitait une intervention directe sur ces pages.

Malgré la richesse des fonctionnalités de LDAP, la mise à jour des informations nécessite normalement l'intervention d'un personnel autorisé en accès sur le serveur. Pour permettre au personnel administratif de le faire lui-même, et dans des conditions d'ergonomie plus simple, (sans aucune connaissance de LDAP) nous avons choisi de créer des pages HTML type "formulaire". Cela a nécessité de développer des scripts en langage PERL aussi bien pour récupérer les informations de saisie que pour traiter les requêtes.

Ainsi, mon travail devait permettre au CRIHAN et à l'INSA de disposer d'un annuaire autorisant des recherches rapides et des mises à jour faciles.

CHAPITRE I : LE CRIHAN

Dans ce chapitre, on va présenter brièvement le lieu du stage, c'est-à-dire le CRIHAN.

I) Le lieu:

Le CRIHAN se situe à Mont Saint Aignan près de Rouen (76). Son adresse est :

CRIHAN
Parc technologique de la Vatine
32, rue Raymon Aron
76130 Mont-Saint-Aignan
Tél : +33 (0)2 35 59 61 59
Fax : +33 (0)2 35 59 61 40

Voici une carte du Parc technologique de la Vatine qui permettra de mieux situer cette association :



II) Le rôle :



Le CRIHAN (Centre de Ressources Informatiques de Haute-Normandie) a été créé en 1992 à l'initiative du Conseil Régional de Haute-Normandie.

C'est une association (loi de 1901) qui comprend 11 permanents dont 5 ingénieurs systèmes et réseaux. Le président en est Dany Vandromme.

Le but de cette association est de "développer des compétences humaines et technologiques à la pointe de la recherche dans les domaines spécifiques d'excellence et reconnues au niveau national et européen afin de contribuer à l'essor économique et culturel de la région, et de renforcer son attraction et son rayonnement".

Possédant des moyens informatiques importants et utilisant des techniques de pointe, le CRIHAN propose, entre autres, un service de modélisation moléculaire pour les laboratoires de Chimie de la région et un service de calcul parallèle basé sur un supercalculateur parallèle qui se situe au troisième rang français par sa puissance de traitement et par sa taille mémoire.

Pour accompagner le projet de réseau régional pour l'enseignement et la recherche, le CRIHAN offre divers services sur SYRHANO¹ et participe à de nombreux projets de développement régional et de promotion des nouvelles technologies d'accès à l'information².

Depuis sa création, une activité de formation s'est développée au sein du CRIHAN. Axée jusqu'alors sur UNIX et Internet, elle s'oriente maintenant vers les technologies avancées et comprend ainsi des modules ATM (réseaux hauts débits), PVM (calculs distribués) et sécurité informatique.

¹ SYstème Réseau de HAute-Normandie

² Les moyens des inforoutes en Haute-Normandie

III) L'équipe :

Comme nous l'avons déjà souligné, l'équipe du CRIHAN se compose de douze permanents dont le président Dany Vandromme.

Les onze permanents sont:

- Monique Boender : Responsable Administrative.
- Béatrice Charton : Ingénieur système, en charge des développements logiciels.
- Jean-Christian Cordier : Responsable des relations extérieures et de la formation.
- Blandine Coupeau : Secrétariat et accueil téléphonique.
- Laurence Dam : Assistante de direction.
- Christèle Domalain : Assistance scientifique aux utilisateurs.
- Edgard Kessy : Support Informatique.
- Natacha Manzano : Ingénieur réseau, en charge des applications de metacomputing
- Alain Massiot : Ingénieur système, en charge des applications graphiques et de l'assistance technique auprès des utilisateurs.
- Hervé Prigent : Responsable Informatique.
- Eric Vilet : Ingénieur système et réseau.

Mon maître de stage était Béatrice Charton mais mon travail a été suivi aussi par Hervé Prigent et Dany Vandromme pour le CRIHAN et par Philippe Wender pour l'INSA.

CHAPITRE II :

LDAP

Lightweight Directory Access Protocol

I) Présentation de LDAP :

LDAP (Lightweight Directory Access Protocol) est le protocole Internet d'annuaire. Il a été développé à l'Université du Michigan à Ann Arbor (Etats-Unis) en collaboration avec l'"Internet Engineering Task Force". Ce protocole LDAP permet de créer et de gérer des annuaires de personnes, de services

Un annuaire est composé de plusieurs entrées ou enregistrements et chaque entrée contient des informations descriptives. Ainsi, un annuaire peut contenir des entrées décrivant des personnes ou du matériel informatique comme des imprimantes par exemple. Un utilisateur peut ainsi utiliser le service d'annuaire pour retrouver le numéro de téléphone d'une personne ou récupérer une liste d'adresse email par exemple.

Les informations sont stockées dans les différents champs de chaque enregistrement. Chaque champ contient un type d'information bien spécifique. Par exemple, ceux servant à enregistrer une personne peuvent être le nom, le prénom, l'adresse email, etc. ...

Donnons un exemple d'enregistrement pour éclairer les choses :

L'enregistrement pour Delphine Fontaine peut être écrit comme suit :

```
cn: Delphine Fontaine
givenName: Delphine
mail: dfontain@crihan.fr
```

Les attributs utilisés sont : `cn` pour "common name", c'est-à-dire le nom utilisé pour nommer la personne (généralement Nom et Prénom), `givenName` pour le prénom, et `mail` pour l'adresse email (courrier électronique).

Remarque : Un attribut peut avoir plusieurs valeurs. Une personne peut donc avoir deux numéros de téléphone ou deux "common name" (le nom et le prénom utilisés couramment puis le diminutif en deuxième `cn`). Ceci donnerait un enregistrement du type :

```
cn: Delphine Fontaine
cn: dede
givenName: Delphine
telephoneNumber: 02 35 01 02 03
telephoneNumber: 02 35 11 22 33
mail: dfontain@crihan.fr
```

Les attributs peuvent aussi contenir des données binaires comme des photos au format JPEG ou la voix d'une personne enregistrée dans un fichier audio.

(cf. chapitre « Création d'une base de données » pour plus d'information sur le format des enregistrements)

Il existe des opérations qui permettent aux clients de rechercher ou de modifier l'annuaire afin de le mettre à jour. Pour faire ces opérations, le client LDAP doit se connecter au serveur LDAP.

Le protocole LDAP prévoit aussi un contrôle d'accès simple. En effet, avant d'effectuer une opération qui modifie l'annuaire, l'utilisateur doit fournir un nom de type dn (appelé rootdn) et un mot de passe. Nous verrons cela dans le chapitre 5, quand on parlera du fichier de configuration.

Le protocole LDAP est basé sur un modèle client-serveur. Les serveurs LDAP fournissent le service d'annuaire et les clients LDAP utilisent le service d'annuaire pour accéder aux données.

II) Comment les serveurs LDAP organisent-ils les annuaires ?

Dans un annuaire, les données sont rangées selon une hiérarchie en arbre (tree hierarchy). En haut de cette hiérarchie se trouve la racine et des branches permettent d'accéder aux enregistrements de l'annuaire.

Généralement, le haut de l'arbre représente un pays, suivi de critères géographiques, politiques, ou autres, puis finalement le nom d'usage (*common name*) de l'enregistrement.

Donnons un exemple :

Pour Dany Vandromme, président du CRIHAN (le CRIHAN étant en France) l'enregistrement serait du type :

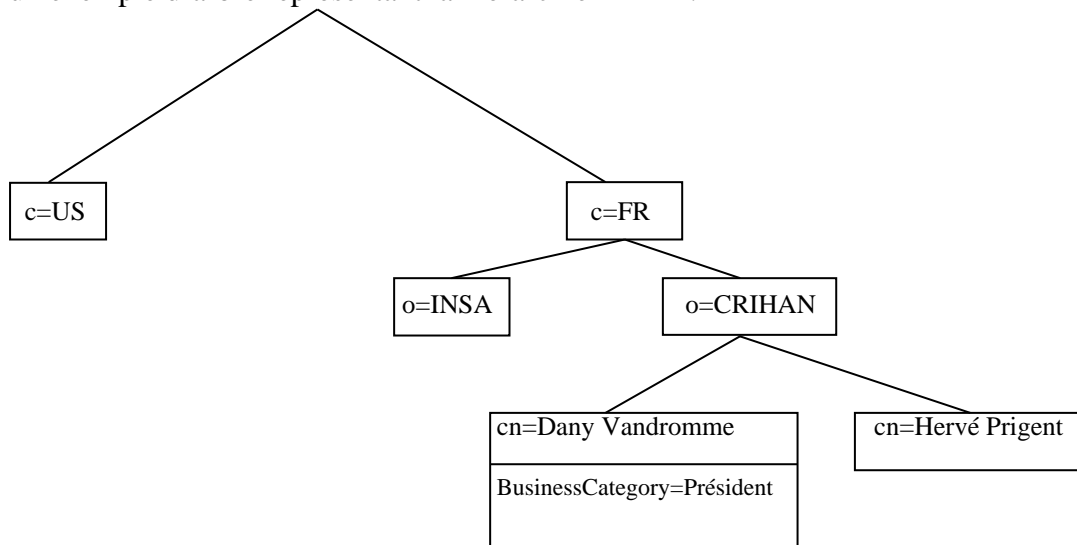
Pays : FR
Organisme : CRIHAN
Nom d'usage : Dany Vandromme

Hervé Prigent qui travaille dans le même organisme aurait un enregistrement du type :

Pays : FR
Organisme : CRIHAN
Nom d'usage : Hervé Prigent

Nous ne tenons pas compte ici du format des enregistrements. Ceci sera détaillé dans le chapitre « Création d'une base de données ».

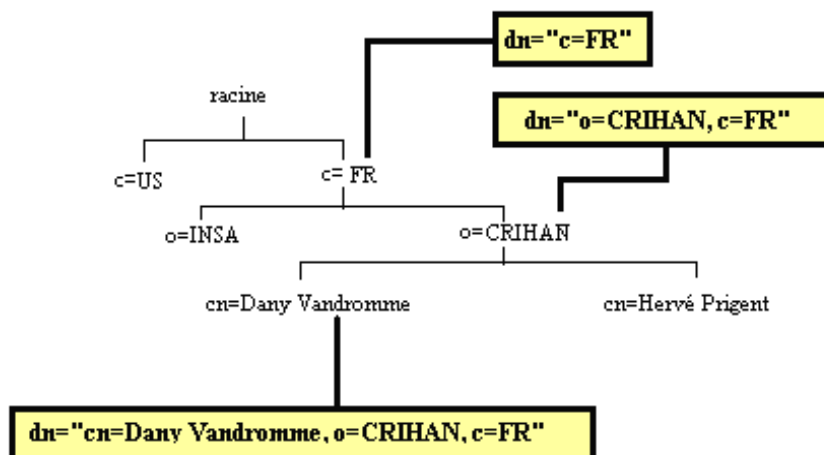
Voici un exemple d'arbre représentant la hiérarchie LDAP :



Chaque enregistrement est uniquement défini par son champ **dn** (*distinguished name*).
Le **dn** pour Dany Vandromme peut être `cn=Dany Vandromme , o=Crihan, c=FR`.

Ici, **cn** représente le nom d'usage de la personne que représente l'enregistrement, **o** est l'organisme auquel appartient la personne, et **c** représente le pays.

Le schéma ci-dessous permet de voir quelques exemples de noms distinctifs "dn" :



A chaque niveau, on a un nom distinctif qui est en fait le chemin pour accéder à un enregistrement à partir de la racine de l'arbre.

Le champs **dn** permet de trouver l'enregistrement correspondant afin d'effectuer l'opération demandée par le client. Ce champ est unique dans la base.

III) Slapd, Slurpd et X.500

LDAP est basé sur le protocole d'annuaire X.500 et son protocole d'interrogation DAP (Directory Access Protocol). DAP était lourd, complexe et difficile à gérer ; alors que LDAP est plus léger, d'où son nom : Lightweight Directory Access Protocol.

Nous ne nous sommes pas intéressés à X.500 puisque nous n'avons fait tourner LDAP qu'à travers le serveur Slapd sans faire tourner X.500.

Slapd (Slapd stand-alone LDAP daemon) est un serveur d'annuaire LDAP qui tourne sur différentes plates-formes UNIX. On peut le connecter au service d'annuaire global LDAP ou le faire fonctionner indépendamment.

Voici quelques-unes de ses caractéristiques :

- *choix d'une base* : Slapd est fourni avec trois types de base de données finales (backend database en anglais). Il y a LDBM, la plus performante, SHELL, une base de données qui permet d'utiliser les commandes Unix dans les recherches mais très limitée, et PASSWD, une simple base de données basée sur un fichier de mots de passes.
- Slapd peut être configuré pour utiliser des bases de données multiples, c'est-à-dire que le serveur Slapd peut répondre à des requêtes concernant différentes parties de l'arbre.
- On peut contrôler les accès à la base.
- Slapd utilise des threads (fil en français, utilisés par les processus multitâches) pour une meilleure performance et choisit la meilleure solution automatiquement selon la plate-forme utilisée.
- On peut configurer facilement le serveur Slapd à l'aide d'un simple fichier de configuration.

Slurpd (stand-alone LDAP update replication daemon) est un démon Unix qui permet à Slapd de dupliquer la base. Il redistribue aux esclaves les changements effectués par le serveur maître. Slapd et Slurpd communiquent par un simple fichier texte (Replication log).

En effet, des serveurs LDAP multiples peuvent contenir des entrées identiques grâce à un mécanisme appelé duplication (*replication* en anglais). Si on configure les serveurs LDAP de telle sorte qu'ils utilisent la duplication, un et un seul serveur gère les modifications de l'annuaire, c'est le fournisseur ou maître (*supplier* en anglais). Les autres serveurs sont les consommateurs ou esclaves (*consumer* en anglais).

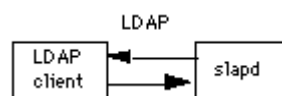
Nous avons choisi le serveur slapd car le CRIHAN n'utilisait pas X.500.

IV) Les différentes configurations possibles :

Voici quelques configurations que l'on peut utiliser. Ces schémas montrent comment les serveurs LDAP (soit slapd soit ldapd) fonctionnent.

LDAP en service local seulement :

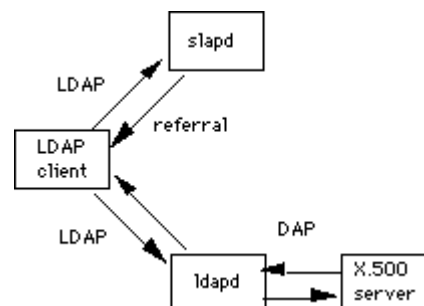
Dans cette configuration, slapd tourne seulement en local, il n'est pas relié aux autres serveurs d'annuaire du monde.



Cette configuration est utilisée quand on ne veut pas se connecter aux annuaires du reste du monde. C'est ce que l'on va utiliser pour le Crihan.

Service local relié aux serveurs X.500 :

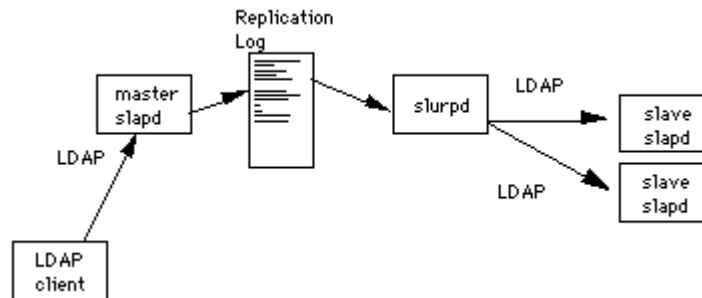
Dans cette configuration, on fait tourner un serveur slapd qui fournit un service d'annuaire local et un serveur ldapd qui fournit un accès aux annuaires X.500.



Cette configuration est utilisée pour garder un contact avec des annuaires X.500 du reste du monde ou pour garder la compatibilité avec une vieille base X.500.

serveur slapd avec base dupliquée :

Le démon slurpd est utilisé pour transmettre les changements du maître slapd aux esclaves slapd. Exemple de configuration maître-esclave :



V) Les différents clients :

On peut utiliser tous les clients LDAP disponibles pour accéder aux enregistrements de l'annuaire et pour les gérer. De plus, des commandes simples sont disponibles dans le répertoire où est installé la base de données. En effet, ces exécutables sont installés en même temps que le serveur.

Le client fourni avec le package ldap :

Les différentes commandes sont expliquées dans le chapitre « Utilisation des différentes commandes ».

Le client Netscape (UNIX, PC, Mac ...) :

Avec Netscape Communicator 4 :

- *Menu Edit/Preferences/Mail and Groups/Directory*: définir un nouvel annuaire (touche New)

On remplit les différents champs :

- *Description*: annuaire Crihan
- *LDAP Server*: ldap.crihan.fr
- *Search Root*: o=crihan,c=FR
- *Port Number*: 389
- *Maximum Number of Hits*: 100

Avec Netscape Communicator 4.5 :

- Ouvrir l'*adress book*
- *Menu File/New/Directory*
- *Description*: annuaire Crihan
- *LDAP Server*: ldap.crihan.fr
- *Search Root*: o=crihan,c=FR
- *Port Number*: 389
- *Maximum Number of Hits*: 100

Ainsi, ce nouveau service d'annuaire va permettre de faire des recherches sur le serveur ldap.crihan.fr (ou grincheux.crihan.fr) qui est le serveur LDAP du Crihan, dans la base dont le suffixe est o=crihan, c=FR (Ceux sont toutes les personnes faisant partie du Crihan).

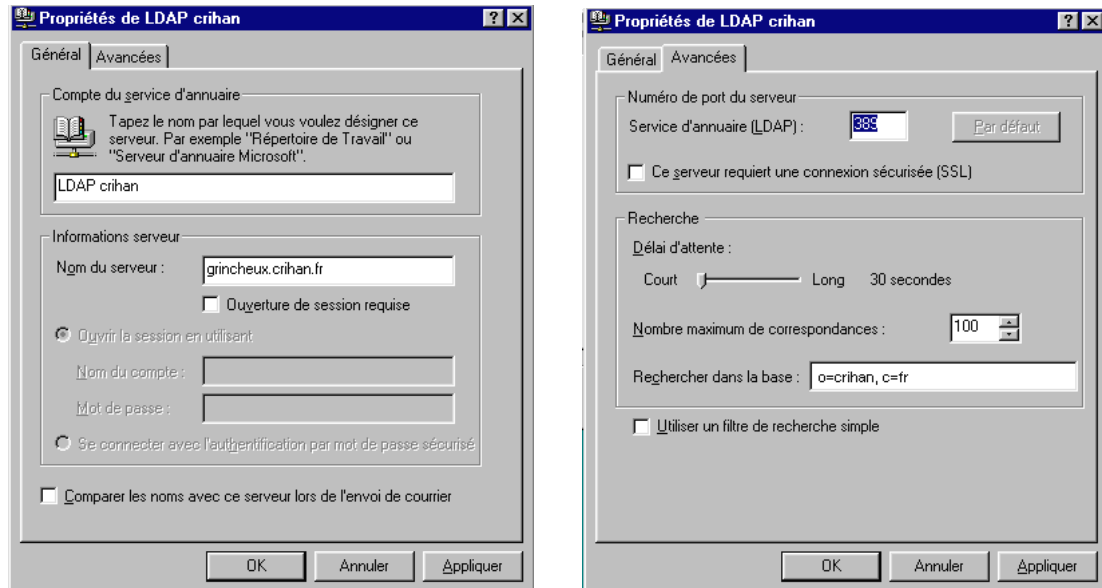
Pour faire les recherches, on ouvre l' « address book », on sélectionne le serveur LDAP que l'on désire utiliser et on remplit le champ avec le nom recherché.

Le client Internet Explorer :

Avec Internet Explorer 4

- Aller dans le menu *Aller à/courrier* pour atteindre Outlook Express
- *Outils/Comptes/*
- Aller dans *Ajouter Service d'annuaire*

On arrive sur la fenêtre de l'assistant. On suit les instructions et on indique le nom du serveur. Pour le serveur du Crihan, ce sera grincheux.crihan.fr ou ldap.crihan.fr. Ensuite, on clique sur le service d'annuaire que l'on vient de créer et on clique sur Propriétés ; ou deux fois sur le nom de l'annuaire. On remplit les différents champs comme suit pour pouvoir faire des recherches dans l'annuaire du Crihan :



Pour faire les recherches, on va dans le menu *Edition/Rechercher* des personnes et on remplit les champs selon notre requête.

Nous pouvons aussi faire les recherches avec d'autres clients LDAP comme LDAPER et Eudora.

CHAPITRE III : L'INSTALLATION

Convention : en gras, ce sont les commandes effectuées et en italique, ce sont les prompts.

J'ai installé LDAP sur une station Sun sous Solaris 2.5 et sur une station Linux. On procède de la même façon dans les deux cas.

Tout d'abord, il faut rapatrier les "packages" nécessaires à l'installation de LDAP. Nous allons décrire l'installation étape par étape pour que les choses soient bien claires :

I) Première étape : Ramener le package LDAP

Nous avons pris la version 3.3 qui est la plus récente. On trouve cette version sur le site gd.tuwien.ac.at et on peut la récupérer par un ftp anonyme. Une fois connecté en tant que ftp anonyme, il faut aller dans le répertoire où se trouve le fichier `ldap-3.3.tar.Z`, c'est-à-dire le répertoire `infosys/network/ldap`. Ainsi, les commandes effectuées sont les suivantes :

```
grincheux.crihan.fr% ftp gd.tuwien.ac.at
Connected to gd.tuwien.ac.at.
220 gd.tuwien.ac.at FTP server (NcFTPD 2.2.2, free educational license) ready.
Name (gd.tuwien.ac.at:bg): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230-You are user #10 of 100 simultaneous users allowed.
230-
230-                               Welcome to
230-                               Goodie Domain Service
230-          Univ. of Technology Vienna, Austria
230-          =====
230-
230- What's in store here ?
230- Yes, Goodies: Freeware, Shareware, PD Software
230-    and lot of hints to goodies at remote sites
230-
230- Access via ftp or http. The paths are the same.
230- Tarring and compression supported.
230-
230- Please read also
230- <http://gd.tuwien.ac.at/no-warranty-note.html> and
230- <http://gd.tuwien.ac.at/priv-policy.html>
230-
230- Sprinzl@edvz.tuwien.ac.at
230-
230 Logged in anonymously.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd infosys/network/ldap
250 "/zauxa/infosys/network/ldap" is new cwd.
```



```
ftp> get ldap-3.3.tar.Z
local: ldap-3.3.tar.Z remote: ldap-3.3.tar.Z
200 PORT command successful.
150 Opening BINARY mode data connection for ldap-3.3.tar.Z (2109567 bytes).

226 Transfer completed.
2109567 bytes received in 34.3 secs (60 Kbytes/sec)
ftp> quit
221 Tks for your visit.
grincheux.crihan.fr%
```

Après avoir récupéré ce fichier ldap-3.3.tar.Z, il faut le décompresser :

```
grincheux.crihan.fr% gunzip ldap-3.3.tar.Z
grincheux.crihan.fr% tar xvf ldap-3.3.tar
```

ou

```
grincheux.crihan.fr% zcat ldap-3.3.tar.Z | tar xvf -
```

Remarque : Il faut d'abord se positionner à l'endroit où l'on veut que le répertoire ldap-3.3 soit situé.

Une fois ce fichier décompressé, nous avons un répertoire ldap-3.3, dont le contenu est :

```
grincheux.crihan.fr% ls ldap-3.3
ANNOUNCEMENT  Make-common      build            include
CHANGES      Make-common.um  clients         libraries
COPYRIGHT     Makefile        contrib         servers
INSTALL       README          doc             tests
```

II) Deuxième étape : Lire les fichiers README et INSTALL

Les fichiers README et INSTALL, qui sont dans le répertoire ldap-3.3 , nous indiquent comment installer LDAP. Ces fichiers sont à lire attentivement. Nous avons donc suivi les instructions de ces deux fichiers pour poursuivre l'installation.

Remarque : Il est aussi recommandé de lire le guide "SLAPD and SLURPD Administrator's Guide", qui se trouve dans le répertoire ldap-3.3/doc/guides. Il est fourni dans le fichier tar récupéré précédemment. Il est en anglais comme les fichiers README et INSTALL.

III) Troisième étape : Modifier les fichiers de configuration.

Cette étape est très importante puisque ce sont ces fichiers qui vont en quelque sorte diriger l'installation propre de LDAP.

Ces fichiers sont le fichier Make-common qui se trouve dans le répertoire ldap-3.3 et le fichier ldapconfig.h.edit qui se trouve dans le répertoire ldap-3.3/include.

Make-common Ce fichier contient des chemins indiquant où seront installés les différents éléments et où se trouve ce dont on a besoin pour l'installation comme les bibliothèques par exemple. Il est nécessaire d'éditer ce fichier pour configurer l'installation à votre convenance.

include/ldapconfig.h.edit Dans ce fichier, nous n'avons besoin de modifier que deux choses : le nom de la machine sur laquelle est situé le serveur LDAP (cette variable est LDAPHOST) et le nom de la base pris par défaut pour les recherches (c'est la variable DEFAULT_BASE).

cf. annexes pour les listing des fichiers Make-common et ldapconfig.h.edit.

Remarque : Le fichier Makefile ne doit pas être modifier.

⇒ Voir le chapitre suivant pour avoir quelques explications sur les fichiers Make-common et ldapconfig.h.edit.

Dans le fichier Make-common, nous avons dû choisir un type de base de données , appelé "low-level database package" en anglais. Nous avons choisi d'utiliser le package GNU DBM (ou GDBM) qui permet la gestion de base de données. Il a donc fallu télécharger les fichiers nécessaires.

IV) Quatrième étape : Téléchargement de gdbm-1.7.3.

Comme pour LDAP, il a fallu faire un ftp anonyme. Nous avons récupéré la version 1.7.3 de GDBM sur le site de Codiciel. Nous avons exécuté les commandes suivantes :

Remarque : On se positionne à l'endroit où l'on veut mettre le répertoire gdbm-1.7.3. Je me mets donc dans le répertoire ldap-3.3 .

```
grincheux.crihan.fr% ftp www.codiciel.fr
Connected to dabe.codiciel.fr.
220 dabe.codiciel.fr FTP server (Version wu-2.4.2-academ[BETA-8])(3) Fri Jan 12
14:23:47 MET 1996) ready.
Name (www.codiciel.fr:bg): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd PUB/GNU
```

```
250-If you have problems downloading and are seeing "Access denied" or
250-"Permission denied", please make sure that you started your FTP client in
250-a directory to which you have write permission.
250-
250-Please note that all files ending in '.gz' are compressed with 'gzip', not
250-with the unix 'compress' program. Get the file below for more info.
250-
250-For a list of mirrors and other ways of getting GNU software, FTP the
250-file /pub/gnu/GNUinfo/FTP from prep.ai.mit.edu or one of it's mirror
250-sites.
250-
250-Please read the file README
250- it was last modified on Tue Jul 9 00:00:00 1996 - 768 days ago
250-Please read the file README-about-.diff-files
250- it was last modified on Thu Mar 20 14:08:00 1997 - 515 days ago
250-Please read the file README-about-.gz-files
250- it was last modified on Tue Jul 9 17:18:00 1996 - 768 days ago
250 CWD command successful.
ftp> get gdbm-1.7.3.tar.gz
local: gdbm-1.7.3.tar.gz remote: gdbm-1.7.3.tar.gz
200 PORT command successful.
150 Opening BINARY mode data connection for gdbm-1.7.3.tar.gz (82379 bytes).
226 Transfer complete.
82379 bytes received in 0.637 secs (1.3e+02 Kbytes/sec)
ftp>quit
```

On décompresse le fichier récupéré.

```
grincheux.crihan.fr% gunzip gdbm-1.7.3.tar.gz
grincheux.crihan.fr% tar xvf gdbm-1.7.3.tar
grincheux.crihan.fr% rm gdbm-1.7.3.tar (ce n'est pas indispensable)
grincheux.crihan.fr% ls
ANNOUNCEMENT      Make-common      build            gdbm-1.7.3      tests
CHANGES          Make-common.um   clients         include
COPYRIGHT         Makefile         contrib         libraries
INSTALL           README           doc             servers
grincheux.crihan.fr%
```

Une fois le répertoire gdbm-1.7.3 décompressé, il faut lire les fichiers README et INSTALL qui se trouvent dans le répertoire gdbm-1.7.3.

V) Cinquième étape : Installer GDBM.

L'installation n'est pas encore faite puisque nous n'avons fait que décompresser le répertoire. Nous suivons les instructions qui sont dans le fichier README : il faut d'abord exécuter la commande `configure` pour que le système utilisé soit reconnu, puis effectuer la commande `make`.

Remarque : Nous avons besoin de la librairie `libgdbm.a` pour pouvoir installer et faire fonctionner notre serveur `slapd`. Nous devons indiquer le chemin de cette librairie dans le fichier `Make-common`. Comme le chemin est indiqué "en dur" dans le `Make-common`, il n'est pas nécessaire d'exécuter la commande "make install". En effet, la génération de la librairie `libgdbm.a` se fait avec seulement la commande `make`.

Voici donc les commandes :



```
grincheux.crihan.fr% cd dbm-1.7.3
grincheux.crihan.fr% ./configure
checking for gcc
checking how to run the C preprocessor
checking for install
checking for ranlib
checking integer size
checking for 64-bit long ints
checking byte ordering
checking for lack of working const
checking for stdlib.h
checking for string.h
checking for sys/file.h
checking for unistd.h
checking for fcntl.h
checking for sys/types.h
checking for memory.h
checking for -ldb
checking for -lndbm
checking for rename
checking for ftruncate
checking for flock
checking for bcopy
checking for fsync
checking for getopt
checking for working alloca.h
checking for alloca
checking for off_t in sys/types.h
checking for st_blksize in struct stat
creating config.status
creating Makefile
creating autoconf.h
grincheux.crihan.fr% make
gcc -c -I. -I. -O dbminit.c
gcc -c -I. -I. -O delete.c
gcc -c -I. -I. -O fetch.c
gcc -c -I. -I. -O store.c
gcc -c -I. -I. -O seq.c
gcc -c -I. -I. -O close.c
gcc -c -I. -I. -O dbmopen.c
gcc -c -I. -I. -O dbmdelete.c
gcc -c -I. -I. -O dbmfetch.c
gcc -c -I. -I. -O dbmstore.c
gcc -c -I. -I. -O dbmseq.c
gcc -c -I. -I. -O dbmclose.c
gcc -c -I. -I. -O dbmdirfno.c
gcc -c -I. -I. -O dbmpagfno.c
gcc -c -I. -I. -O dbmrdoonly.c
gcc -c -I. -I. -O gdbmopen.c
gcc -c -I. -I. -O gdbmdelete.c
gcc -c -I. -I. -O gdbmfetch.c
gcc -c -I. -I. -O gdbmstore.c
gcc -c -I. -I. -O gdbmclose.c
gcc -c -I. -I. -O gdbmreorg.c
gcc -c -I. -I. -O gdbmseq.c
gcc -c -I. -I. -O gdbmsync.c
gcc -c -I. -I. -O gdbmerrno.c
gcc -c -I. -I. -O gdbmexists.c
gcc -c -I. -I. -O gdbmsetopt.c
gcc -c -I. -I. -O bucket.c
gcc -c -I. -I. -O falloc.c
gcc -c -I. -I. -O findkey.c
gcc -c -I. -I. -O global.c
gcc -c -I. -I. -O hash.c
gcc -c -I. -I. -O update.c
gcc -c -I. -I. -O version.c
rm -f gdbm.h
cp ./gdbm.proto gdbm.h
chmod +w gdbm.h
grep _ ./gdbmerrno.h >> gdbm.h
cat ./gdbm.proto2 >> gdbm.h
```

```
chmod -w gdbm.h
rm -f libgdbm.a
ar q libgdbm.a dbmdelete.o dbmfetch.o dbmstore.o dbmseq.o dbmclose.o dbmdirfno.o dbmpagfno.o dbmrdonly.o gdbmopen.o
dbmdelete.o dbmfetch.o gdbmstore.o gdbmclose.o gdbmreorg.o gdbmseq.o gdbmsync.o
gdbmerrno.o gdbmexists.o gdbmsetopt.o bucket.o falloc.o findkey.o global.o hash.o update.o
version.o
ar: creating libgdbm.a
ranlib libgdbm.a
grincheux.crihan.fr% ls
COPYING      dbmdelete.c      fetch.c          gdbmopen.c      seq.o
ChangeLog    dbmdelete.o      fetch.o          gdbmopen.o      store.c
INSTALL      dbmdirfno.c      findkey.c        gdbmreorg.c     store.o
Makefile     dbmdirfno.o      findkey.o        gdbmreorg.o     systems.h
Makefile.in  dbmfetch.c       gdbm.3           gdbmseq.c       tdbm
NEWS         dbmfetch.o       gdbm.h           gdbmseq.o       testdbm
README       dbmdelete.c      gdbm.info        gdbmsetopt.c    testdbm.c
alloca.c     dbmdelete.o      gdbm.proto       gdbmsetopt.o    testdbm.o
autoconf.h   dbmopen.c        gdbm.proto2      gdbmstore.c     testgdbm
autoconf.h.in dbmopen.o        gdbm.texinfo     gdbmstore.o     testgdbm.c
bucket.c     dbmpagfno.c      gdbmclose.c      gdbmsync.c      testgdbm.o
bucket.o     dbmpagfno.o      gdbmclose.o      gdbmsync.o      testndbm
close.c      dbmrdonly.c      gdbmconst.h      getopt.c        testndbm.c
close.o      dbmrdonly.o      gdbmdefs.h       getopt.h        testndbm.o
config.status dbmseq.c         gdbmdelete.c     global.c        tndbm
configure    dbmseq.o         gdbmdelete.o     global.o        tndbm.o
configure.in dbmstore.c       gdbmerrno.c      hash.c          update.c
conv2gdbm    dbmstore.o       gdbmerrno.h      hash.o          update.o
conv2gdbm.c  delete.c         gdbmerrno.o      libgdbm.a       version.c
conv2gdbm.o  delete.o         gdbmexists.c     ndbm.h          version.o
dbm.h        extern.h         gdbmexists.o     proto.h
dbmclose.c   falloc.c         gdbmfetch.c      sampl.cc
dbmclose.o   falloc.o         gdbmfetch.o      seq.c
grincheux.crihan.fr%
```

Maintenant, tout est prêt pour faire l'installation proprement dite de LDAP.

VI) Sixième étape : Make.

On revient dans le répertoire ldap-3.3 :

```
grincheux.crihan.fr% cd /usr/local/home/dfontain/ldap-3.3
```

Le Makefile permet d'exécuter les commandes suivantes :

- make veryclean : pour tout nettoyer dans le répertoire (effacer les bibliothèques, ...)
- make : pour créer les bibliothèques et préparer l'installation
- make install : pour installer le logiciel (software) LDAP en créant les répertoires et en copiant les fichiers à l'endroit indiqué dans le fichier Make-common.

Nous faisons donc ici :

```
grincheux.crihan.fr% make
```

Il faut faire bien attention à ce qui s'inscrit sur l'écran, pour voir s'il n'y a pas d'erreur dans le déroulement de cette commande.

Problèmes rencontrés lors du make :

voir le chapitre « configuration utilisée et problèmes rencontrés ».

Remarque :

Si on utilise Solaris2.x et que l'on utilise un package externe de base de données comme db et gdbm, il est important de compiler le package avec l'option -DREENTRANT.

En fait, cette option est mise automatiquement puisque lors du "make", le système est reconnu automatiquement.

Pendant le "make", il fait un "make-platform" où il reconnaît l'OS (le système d'exploitation) où il fait le "make".

(Pour connaître notre machine, on tape la commande :

```
grincheux.crihan.fr% uname -a
SunOS grincheux.crihan.fr 5.5 Generic sun4m sparcsun4m SPARCstation-5
grincheux.crihan.fr% )
```

Voici un extrait du fichier ldap-3.3/Makefile :

```
.make-platform:
....
    if [ -z "$$UNAME" ]; then \
        echo "unknown platform (no $$UNAME or /usr/bin/uname)"; \
        echo "see the file build/PORTS for more information."; \
        exit 1; \
    else \
        OS=`$$UNAME -s` ; OSRELEASE=`$$UNAME -r` ; \
        OSVERSION=`$$UNAME -v` ; \
        case $$OS in \
            SunOS) \
                if [ $$OSRELEASE -gt "5" -o $$OSRELEASE -lt "4" ]; then \
                    echo "SunOS release $$OSRELEASE unknown..."; exit 1; \
                fi; \
                if [ $$OSRELEASE -ge "5" ]; then \
                    PLATFORM="sunos5"; \
                else \
                    PLATFORM="sunos4"; \
                fi; \
                ;; \
            Linux) \
                PLATFORM="linux" \
                ;; \
        esac
....
```

Ainsi, nous voyons qu'il reconnaît la "plate-forme", ici ce sera Sunos5. Puis, il va aller dans le répertoire ldap-3.3/build/platforms où il y a un répertoire par type de plate-forme. Ensuite, il exécutera le Makefile dans le répertoire correspondant à notre machine. Dans ce Makefile, un "Make-platform" sera appelé. Voyons comment se présente le fichier "Make-platform" pour une station sunos5 et pour linux, tous deux munis du compilateur gcc.

```

grincheux.crihan.fr% cd /usr/local/home/dfontain/ldap-3.3/build/platforms
grincheux.crihan.fr% ls
Make-setup      hpux-gcc          nextstep-cc      sunos5-gcc
Makefile        irix-cc           osf1-cc          ultrix-cc
aix-cc          irix-gcc          sco-cc           ultrix-gcc
aix-gcc         linux-gcc         sco-gcc          vms
freebsd-gcc     ncr-mp-ras-2-cc  sunos4-cc
hpux-c89        netbsd-cc         sunos4-gcc
hpux-cc         netbsd-gcc        sunos5-cc
grincheux.crihan.fr% cd sunos5-gcc
grincheux.crihan.fr% ls
Make-platform  Makefile
grincheux.crihan.fr% more Make-platform
#
# LDAP SunOS5 GNU C Make-platform file
#
#
# add any platform-specific overrides below here
#
# -----
# you will probably not need to edit anything below this point
# -----
CC          = gcc

# ranlib not needed under SunOS5
RANLIB     = ""

# give full path to hostname since it may not be in user's path
HOSTNAME=/usr/ucb/hostname

# don't count on /usr/ucb/install being present or first in path
INSTALL=$(LDAPSRC)/build/install.sh

PLATFORMCFLAGS= -Dsunos5 -D_REENTRANT
PLATFORMLIBS= -lsocket -lnsl -lgen
THREADS= -DTHREAD_SUNOS5_LWP
THREADSLIB=-lthread
grincheux.crihan.fr% cd ..
grincheux.crihan.fr% cd linux-gcc
grincheux.crihan.fr% ls
Make-platform  Makefile
grincheux.crihan.fr% more Make-platform
#
# LDAP Linux GNU C Make-platform file
#
#
# add any platform-specific overrides below here
#
# -----
# you will probably not need to edit anything below this point
# -----
CC          = gcc
RANLIB     = "ranlib"

PLATFORMCFLAGS= -Dlinux
grincheux.crihan.fr%

```

Nous voyons bien que dans le fichier "Make-platform" du répertoire sunos5 (équivalent de Solaris2.5), il met l'option -D_REENTRANT dans une variable qui sera utilisée par le Makefile général (qui fait appel à tous les Makefiles de tous les sous répertoires de ldap-3.3). Alors que pour linux, il n'a pas besoin de cette option alors il ne la met pas.

VII) Septième étape : Make test.

Avant que le serveur soit installé, il faut tester son fonctionnement. Pour cela, des tests ont été fournis avec le package ldap-3.3.

Les programmes tests sont dans le répertoire ldap-3.3/tests et il suffit d'exécuter la commande "make test" dans ce répertoire pour exécuter les sept tests existants.

```
grincheux.crihan.fr% pwd
/usr/local/home/dfontain/ldap-3.3
grincheux.crihan.fr% cd tests
grincheux.crihan.fr% make
Initiating LDAP tests...
>>>> Executing all LDAP tests...
>>>> Starting ./scripts/test001-ldif2ldb ...
Cleaning up in ./test-db...
Running ldif2ldb to build slapd database...
Starting slapd on TCP/IP port 9009...
Using ldapsearch to retrieve all the entries...
Waiting 5 seconds for slapd to start...
Comparing retrieved entries to LDIF file used to create database
>>>> Test succeeded
>>>> ./scripts/test001-ldif2ldb completed OK.

>>>> Starting ./scripts/test002-populate ...
Cleaning up in ./test-db...
Starting slapd on TCP/IP port 9009...
Using ldapsearch to check that slapd is running...
Using ldapmodify to populate the database...
Using ldapsearch to read all the entries...
Comparing retrieved entries to LDIF file used to create database
>>>> Test succeeded
>>>> ./scripts/test002-populate completed OK.

.....
```

Ceci permet de voir si la compilation s'est déroulée correctement.

VIII) Huitième étape : Make install.

Pour exécuter cette commande, il faut se connecter en tant que root.
Les commandes seront donc :

```
grincheux.crihan.fr% su
# make install
```

Si la commande "make" s'est déroulée correctement, le make install se passe bien.

Remarque : Cette commande peut être faite sans être connecté en tant que root. Dans ce cas, les fichiers devront être installés dans un répertoire accessible à l'utilisateur qui installe slapd. Cependant, cet utilisateur ne pourra pas exécuter la commande "slapd" (pour démarrer le serveur) car il faut que cet exécutable appartienne à root.

Les droits sur slapd sont :

```
grincheux.crihan.fr% ls -al slapd
-rwxr-xr-x  1 dfontain  crihan  288748 Sep  8 14:12 slapd
grincheux.crihan.fr%
```

Il faudrait alors changer les droits sur cet exécutable avec les commandes suivantes en étant connecté en tant que root :

```
grincheux.crihan.fr# chown root slapd
grincheux.crihan.fr# chmod 4755 slapd
```

Les droits sur "slapd" seraient :

```
grincheux.crihan.fr% ls -al slapd
-rwsr-xr-x  1 root  crihan  288748 Sep  8 14:15 slapd
grincheux.crihan.fr%
```

Ainsi, slapd est exécutable sans que le "make install" n'ait été fait en tant que root. C'était mon cas sur grincheux.crihan.fr. En effet, lors de son exécution, slapd se comporte comme si c'était root qui l'avait lancé.

L'installation faite, il faut maintenant apprendre à utiliser les différentes commandes. Nous avons vu dans ce chapitre que l'étape la plus difficile était la sixième, avec l'exécution de la commande make. En effet, si les chemins des bibliothèques ne sont pas bien indiqués, ou si des liens ne sont pas trouvés, le make ne se fait pas correctement. C'est pour cela que la configuration du Make-common doit être faite avec la plus grande attention. Nous allons donc voir plus en détail ce fichier dans le chapitre suivant.

CHAPITRE IV :

LE FICHER MAKE-COMMON

Ce fichier est un fichier très important pour l'installation. Nous allons détailler le fichier Make-common en expliquant brièvement les changements à effectuer. Le chapitre suivant donnera la configuration choisie pour installer LDAP au CRIHAN.

I) Les chemins d'installation de LDAP :

Tout sera installé dans le répertoire INSTROOT dont il faudra donner la valeur. Dans ce répertoire, il y aura plusieurs sous-répertoires :

- etc : il va contenir les fichiers de configuration, l'exécutable slapd etc...
- include : il va contenir les fichiers dont l'extension est « *.h »
- lib : ce répertoire va contenir les bibliothèques
- man : man va contenir tous les fichiers man concernant LDAP et les différentes commandes disponibles
- bin : il va contenir les binaires des commandes : ldapadd, ldapdelete, ldapsearch ...

Les chemins pour accéder à ces répertoires sont mis dans des variables : ETCDIR, INCLUDEDIR, LIBDIR, MANDIR, BINDIR. Ces variables ne sont pas à changer sauf si l'on veut mettre les fichiers man dans un autre répertoire que man par exemple.

INSTROOT a pour valeur par défaut /usr/local.

II) Le package KERBEROS :

Si les données que l'on mettra dans la base sont confidentielles et accessibles que par certaines personnes, un système de chiffrement est possible. Ainsi, le package KERBEROS peut être installé et le passage concernant ce type d'authentification dans le Make-common doit être décommenté. Les chemins doivent être changés selon l'endroit où se situent les bibliothèques nécessaires.

III) Le package ISODE :

ISODE est indispensable uniquement pour construire le serveur LDAP qui assure la liaison entre LDAP et X500.

Si l'on possède X500, il faut installer le package ISODE, décommenter la ligne avec HAVEISODE et la partie correspondant à la version d'ISODE installée dans le Make-common.

IV) La partie concernant Slapd :

Pour construire slapd, il faut décommenter la ligne "MAKESLAPD=yes" et choisir la ou les types de base de données finales (SLAPD_BACKENDS).

Trois bases de données finales sont disponibles :

- `-DLDAPE_LDBM` : C'est une base de données très performante qui permet de traiter des millions d'entrées (ou enregistrements).
- `-DLDAPE_PASSWD` : Cette base de données permet de faire seulement des recherches qui peuvent pointer sur le fichier `/etc/passwd` mais elle n'est pas très performante.
- `-DLDAPE_SHELL` : Cette base de données permet d'utiliser des commandes UNIX dans les recherches comme `grep`, `awk`

Ainsi, si on veut permettre l'utilisation des bases LDBM et PASSWD, il faut modifier la ligne définissant SLAPD_BACKENDS pour avoir :

```
SLAPD_BACKENDS = -DLDAPE_LDBM -DLDAPE_PASSWD.
```

Remarque : On peut laisser les trois, cela ne veut pas dire qu'on les utilisera forcément toutes les trois.

Si on a mis `-DLDAPE_LDBM` dans la variable SLAPD_BACKENDS, il faut choisir un package de base. Ce sont des ensembles de fichiers facilitant la gestion de base de données. Ce package permet de formater les données en binaire et de les ordonner pour que LDAP y accède plus rapidement.

On a le choix entre quatre :

- Le package Berkeley db btree :
- Le package Berkeley db hash
- Le package gnu dbm (gdbm)
- Le standard Unix : ndbm

Le dernier est fourni avec le système Unix. Sinon, il faut récupérer le package avant d'installer LDAP.

Après avoir fait notre choix, il faut décommenter la partie correspondante du Make-common et indiquer les bons chemins pour accéder aux bibliothèques.

Il est conseillé d'utiliser d'autres packages que ndbm pour une meilleure performance.

L'option THREADS est automatiquement configurée dans le fichier Make-platform (que nous avons vu au chapitre précédent) donc nous n'avons pas besoin de décommenter ces lignes ; sauf si l'on veut utiliser un package qui n'est pas celui utilisé par défaut.

V) Le client xax500 :

On ne s'intéressera à cette section que si le source xax500 a été récupéré et mis dans le répertoire "clients". Si c'est le cas, il faut décommenter les lignes et indiquer les bons chemins pour accéder aux bibliothèques et aux fichiers include.

VI) Les caractères :

Pour pouvoir utiliser des caractères accentués, il faut décommenter la ligne
`LIBLDAP_CHARSETS=-DLLDAP_CHARSET_8859="88591"`

88591 représente le jeu de caractères ISO 8859-1 (caractères ISO-Latin). Si on utilise ISO 8859-4, il faut indiquer 88594 à la place de 88591.

VII) Les options :

Il suffit de lire les commentaires dans le fichier "Make-common" et de décommenter les options que l'on veut disponibles.

Par exemple :

Si l'on veut pouvoir utiliser le débogage, il faut décommenter la ligne :

```
LDAP_DEBUG=-DLLDAP_DEBUG.
```

Cette option est conseillée.

Une fois les fichiers Make-common et ldapconfig.h.edit configurés, l'installation pourra se faire, normalement sans problème mais ce n'est pas toujours le cas !

CHAPITRE V :

CONFIGURATION UTILISEE ET PROBLEMES RENCONTRES

I) MATERIEL UTILISE :

J'ai fait l'installation sur une station Sun. La machine se nomme grincheux.crihan.fr .
On peut voir les caractéristique de cette machine avec la commande "uname -a" :

```
grincheux.crihan.fr% uname -a  
SunOS grincheux.crihan.fr 5.5 Generic sun4m sparcs SUNW,SPARCstation-5  
grincheux.crihan.fr%
```

Pour l'INSA, l'installation s'est faite aussi sur un Sun sous Solaris 2.6. La machine s'appelle intranet :

```
intranet% uname -a  
SunOS intranet 5.6 Generic_105181-03 sun4u sparcs SUNW,Ultra-5_10  
intranet%
```

L'installation avait été commencée sur une machine possédant Linux mais elle a été abandonné pour être faite sur une station Sun afin de permettre l'accès à l'annuaire par l'Intranet.

Le principe d'installation est identique sous Unix ou sous Linux. Mais des outils sont nécessaires pour installer LDAP :

- Tar (qui est standard Unix) et gunzip pour décompresser ldap-3.3.tar.Z et gdbm-1.7.3.tar.gz
- Le compilateur gcc bien installé (ou tout autre compilateur C).
- la commande make (qui est standard à Unix) dont le chemin doit être mis dans la variable d'environnement PATH.

II) CONFIGURATION :

- Modification du fichier include/ldapconfig.h.edit :

Dans ce fichier, il a juste fallu indiquer le nom de la machine sur laquelle se trouvait le serveur LDAP : ldap.crihan.fr (c'est un alias pour grincheux.crihan.fr).

Puis, j'ai indiqué le nom de la base prise par défaut pour les recherches. Comme on est en France, on indique c=FR (c pour country ou pays) et comme le stage se déroulait au CRIHAN, on met o=crihan (o pour organization).



Pour que les choses soient plus claires, Voici l'extrait du fichier include/ldapconfig.h.edit qui a été modifié:

```
/*
 * SHARED DEFINITIONS - things you should change
 */
/* default ldap host */
#define LDAPHOST "ldap.crihan.fr"
/* default place to start searching */
#define DEFAULT_BASE "o=crihan, c=FR"
```

- Modification du fichier Make-common :

Voici les parties du fichier qui ont été changées :

```
# by default, everything is installed below INSTROOT
# servers, config files, etc. are put in ETCDIR
# include files get put in INCLUDEDIR
# libraries are put in LIBDIR
# man pages are put under MANDIR
# programs end-users will run are put in BINDIR
#
INSTROOT=/usr/local/home/dfontain/ldap-3.3
ETCDIR= $(INSTROOT)/etc
INCLUDEDIR= $(INSTROOT)/include
LIBDIR= $(INSTROOT)/lib
MANDIR= $(INSTROOT)/man
BINDIR= $(INSTROOT)/bin
```

En fait, dans cette partie, il n'y a que la variable INSTROOT qui a été modifiée. Comme pendant mon stage, il a fallu apprendre à se servir de LDAP, l'installation s'est faite dans mon répertoire principal : dfontain. Je n'avais pas les droits root pour pouvoir l'installer dans /usr/local/.

Ainsi, tous les répertoires sont installés directement dans le répertoire ldap-3.3 qui se trouve dans mon répertoire principal : /usr/local/home/dfontain/ldap-3.3.

Je n'ai pas décommenté la partie concernant le package KERBEROS puisque la base du CRIHAN doit être mise sur Internet avec l'accord des personnes figurant dans l'annuaire. Ainsi, une méthode d'authentification n'est pas nécessaire.

La partie concernant le package ISODE ne nous concernait pas non plus puisque le CRIHAN n'utilisait pas X500. Donc, on laisse les commentaires.

```
#####
## If you don't want to run slapd, skip this section. ##
#####
#
# To build slapd (the stand-alone ldap daemon), uncomment the MAKESLAPD
# line and select the SLAPD_BACKENDS you want to use. If you enable the
```

```
# LDBM backend, also select one of the LDBM backends.
MAKESLAPD= yes
#
# remove the defines for backends you don't want to enable
SLAPD_BACKENDS= -DLDBM -DLDBM_SHELL -DLDBM_PASSWD
#
# If you have included -DLDBM in the SLAPD_BACKENDS line you need
# to specify which low-level database package to use. There are
# four choices: Berkeley db b-tree, Berkeley db hash, GNU dbm, or ndbm.
#
# berkeley db btree package
#LDBM_BACKEND=-DLDBM_USE_DBBTREE
#LDBM_INCLUDE=-I/usr/local/db/include
#LDBMLIB=-ldb
# berkeley db hash package
#LDBM_BACKEND=-DLDBM_USE_DBHASH
#LDBM_INCLUDE=-I/usr/local/db/include
#LDBMLIB=-ldb
# gnu dbm (gdbm)
LDBM_BACKEND=-DLDBM_USE_GDBM
LDBM_INCLUDE=-I/usr/local/home/dfontain/ldap-3.3/gdbm-1.7.3
LDBMLIB=-lgdbm
# standard unix ndbm
#LDBM_BACKEND=-DLDBM_USE_NDBM
#
```

Cette partie est importante pour nous car le serveur slapd va permettre de gérer la base et de supprimer, ajouter ou modifier les enregistrements dans la base afin de mettre à jour l'annuaire.

Nous avons laissé les trois types de base finale ou *backends* :

```
SLAPD_BACKENDS= -DLDBM -DLDBM_SHELL -DLDBM_PASSWD
```

Cependant, nous n'utiliserons que `ldbm` car c'est la base de données la plus performante.

Pour utiliser `ldbm` (`-DLDBM`), il nous faut choisir un package avec des éléments pour gérer une base de données. Nous avons choisi de prendre le package GNU dbm car `ndbm` (qui est intégré au système Unix) est insuffisant.

Ainsi, on décommente les trois lignes concernant GNU dbm, et on précise le chemin de la bibliothèque de ce package, à la ligne `LDBM_INCLUDE`.

Remarque : Comme nous l'avons vu dans un des chapitres précédents, une fois le package `gdbm-1.7.3` rapatrié, il faut exécuter quelques commandes afin de créer la librairie dont on a besoin :

Une fois dans le répertoire `gdbm-1.7.3`, il suffit de faire :

```
% ./configure
```

Cette commande permet de reconnaître le système utilisé et de configurer certaines variables.

```
% make
```

Le `make` permet de construire la librairie `libgdbm.a`, celle dont on avait besoin pour installer LDAP. Il n'est pas nécessaire de faire "make install" puisque la librairie est la seule chose dont nous avons besoin. Elle est située directement dans le répertoire `gdbm-1.7.3`. Cependant, si nous avions voulu mettre la librairie dans le répertoire habituel : `/usr/lib`, il aurait fallu faire l'installation complètement ("make install").

Ici, le chemin `LDBM_INCLUDE` est donc `/usr/local/home/dfontain/ldap-3.3/gdbm-1.7.3`.

voir le chapitre installation pour le résultat des commandes.

Dans la partie concernant les caractères, on ne décommente qu'une seule ligne :
LIBLDAP_CHARSETS=-DLDA_CHARSET_8859="88591"
Ceci permet de convertir les caractères au format ISO 8859-1 (voir le problème concernant les accents dans un chapitre plus loin). Nous avons donc :

```
#####  
## If you don't want to do auto-translation of character sets, skip this ##  
#####  
#  
# Otherwise, uncomment this line and set the following options.  
#STR_TRANSLATION=-DSTR_TRANSLATION  
#  
# remove the defines for LDAP client library T.61 character translation  
# you do not need. If you use LDAP_CHARSET_8859, replace the '1' in "88591"  
# with the number of the particular character set you use. E.g., use "88594"  
# if you use the ISO 8859-4 chracter set.  
LIBLDAP_CHARSETS=-DLDA_CHARSET_8859="88591"  
#  
# uncomment one these lines to enable automatic T.61 translation by default  
#LIBLDAP_DEF_CHARSET=-DLDA_DEFAULT_CHARSET=LDAP_CHARSET_8859
```

La dernière partie du fichier Make-common concerne les options. Nous laissons cette partie comme elle était.

III) PROBLEMES RENCONTRES :

Après avoir modifié les fichiers Make-common et ldapconfig.h.edit, j'ai fait le make dans le répertoire ldap-3.3. Le make ne s'est pas effectué sans problème dès la première fois. Voici donc un récapitulatif des différents problèmes que j'ai eu et les solutions à ces problèmes, solutions parfois très simples mais il suffisait d'y penser !!

❖ En exécutant la commande "make" sur une station SunOS 5.5, le message suivant peut s'afficher :

```
grincheux.crihan.fr% make
```

```
.....  
gcc -I../include -Dsunos5 -D_REENTRANT -DLDA_DEBUG -DLDA_REFERRALS  
-DLDA_CHARSET_8859="88591" -DLDA_LDBM -DLDA_SHELL -DLDA_PASSWD -  
DLDBM_USE_NDBM -c version.c  
gcc -o in.xfingerd main.o version.o \  
-L../libraries -lldap -llber -lsocket -lnsl -lgen  
  
cd gopher; make all  
gcc -I../include -Dsunos5 -D_REENTRANT -DLDA_DEBUG -DLDA_REFERRALS  
-DLDA_CHARSET_8859="88591" -DLDA_LDBM -DLDA_SHELL -DLDA_PASSWD -  
DLDBM_USE_NDBM -c go500gw.c  
go500gw.c:66: `MAXHOSTNAMELEN' undeclared here (not in a function)  
*** Error code 1  
make: Fatal error: Command failed for target `go500gw.o'  
Current working directory /usr/local/home/dfontain/ldap-3.3/clients/gopher  
*** Error code 1  
make: Fatal error: Command failed for target `all'  
Current working directory /usr/local/home/dfontain/ldap-3.3/clients  
*** Error code 1  
make: Fatal error: Command failed for target `all'  
grincheux.crihan.fr%
```

En fait, il manque un `#define` dans le fichier `go500gw.c` et c'est la même chose pour le fichier `go500.c`. C'est une ligne contenue normalement dans des fichiers include systèmes. Plutôt que de changer la ligne dans les fichiers systèmes, on modifie les deux fichiers concernés. Pour connaître la ligne à ajouter, on regarde les fichiers include système d'une autre machine.

Solution :

On rajoute la ligne `#define MAXHOSTNAMELEN 256` au début de chaque fichier après les `#include`.

Les deux fichiers se trouvent dans le répertoire `ldap-3.3/clients/gopher`.

Le début du fichier `go500gw.c` devient donc :

```
grincheux.crihan.fr% more go500gw.c
.....
#include <sys/resource.h>
#include <sys/wait.h>
#include <signal.h>
#ifdef aix
#include <sys/select.h>
#endif /* aix */
#include "portable.h"
#include "ldapconfig.h"

#ifdef USE_SYSCONF
#include <unistd.h>
#endif /* USE_SYSCONF */

#define MAXHOSTNAMELEN 256

int    debug;
int    dosyslog;
int    inetd;

.....
```

On fait la même chose avec le fichier `go500.c`.

❖ Une erreur fréquente lors du "make" est :

```
ld: fatal: library -lgdbm: not found
```

Le programme ne trouve pas la librairie `lgdbm`. C'est en fait la librairie créée lors du "make" effectué dans le répertoire `gdbm-1.7.3`. La librairie n'est pas trouvée car on n'a pas fait le "make install", on avait juste fait le "make" afin de créer la librairie.

Solution :

Il suffit de rajouter le chemin pour accéder à cette librairie dans la variable d'environnement `LD_LIBRARY_PATH`.

Dans mon cas, la variable `LD_LIBRARY_PATH` n'était pas définie alors je la définis en lui indiquant le chemin de la librairie `lgdbm` qui se nomme `libgdbm.a` et qui se trouve dans le répertoire `ldap-3.3/gdbm-1.7.3` :

```
grincheux.crihan.fr% echo $LD_LIBRARY_PATH
LD_LIBRARY_PATH: Undefined variable
grincheux.crihan.fr% setenv LD_LIBRARY_PATH /usr/local/home/dfontain/ldap-3.3/gdbm-1.7.3
```

grincheux.crihan.fr%

Si d'autres bibliothèques n'étaient pas trouvées, il faudrait indiquer leur chemin d'accès dans la variable LD_LIBRARY_PATH (chaque répertoire étant séparé par ":").

- ❖ Quelquefois, le make ne peut pas s'exécuter car le compilateur gcc est introuvable.

Solution :

Dans ce cas, il faut vérifier la variable d'environnement PATH. Cette variable doit contenir le chemin d'accès au compilateur gcc.

Ainsi, sur la machine que j'utilisais, le chemin était /opt/gnu/bin/gcc. J'ai donc effectué la commande suivante :

```
grincheux.crihan.fr% setenv PATH ${PATH}:/opt/gnu/bin  
grincheux.crihan.fr%
```

S'il ne trouve pas certaines commandes, il faut lui en indiquer le chemin dans la variable PATH.

- ❖ Une fois le make terminé avec succès, il faut exécuter la commande "make install". Lors de l'exécution de cette commande, des messages d'erreur peuvent s'afficher mais ils sont ignorés. Ainsi, on voit apparaître des messages tels que :

```
Error code 2 (ignored)
```

Ces messages ne doivent pas nous inquiéter puisque les erreurs sont ignorées. Si on refait la commande "make install", on remarque l'absence de ces erreurs.

- ❖ Une remarque importante : Il faut toujours exécuter la commande "**make veryclean**" avant de refaire un "make", sinon, des erreurs peuvent apparaître lors de l'exécution de la commande "make test" ou lors de la création d'une base.

Ce sont des erreurs telles que :

```
grincheux.crihan.fr% pwd  
/usr/local/home/dfontain/ldap-3.3  
grincheux.crihan.fr% cd tests  
grincheux.crihan.fr% make  
Initiating LDAP tests...  
>>>> Executing all LDAP tests...  
>>>> Starting ./scripts/test001-ldif2ldb...  
Cleaning up in ./test-db...  
Running ldif2ldb to build slapd database...  
id2entry ldbm_store: No space left on device  
stopping: child exited with status 1  
ldif2ldb failed!  
>>>> ./scripts/test001-ldif2ldb failed (exit 1)  
*** Error code 1  
make: Fatal error: Command failed for target `all'  
grincheux.crihan.fr%
```

ou

```
grincheux.crihan.fr% ldif2ldb -i base_ldif -f fic.conf
idl_insert_key: No space left on device
stopping: child exited with status 1
grincheux.crihan.fr%
```

On fait donc un "make veryclean" avant d'exécuter "make" et "make install".

❖ Un problème à l'exécution du make install pour la deuxième fois est survenu.

```
cd back-shell; make all
`libback-shell.a' is up to date.

`libbackends.a' is up to date.
`slapd' is up to date.
uncomment the HAVEISODE=yes line in the Make-common file to build edb2ldif
../../build/install.sh -c -m 755 slapd /usr/local/home/dfontain/ldap-3.3/etc
cp: cannot create /usr/local/home/dfontain/ldap-3.3/etc/slapd: Permission denied
chmod: WARNING: can't change /usr/local/home/dfontain/ldap-3.3/etc/slapd
*** Error code 1
make: Fatal error: Command failed for target `/slapd'
Current working directory /usr/local/home/dfontain/ldap-3.3/servers/slapd
*** Error code 1
make: Fatal error: Command failed for target `install'
Current working directory /usr/local/home/dfontain/ldap-3.3/servers
*** Error code 1
make: Fatal error: Command failed for target `install'
```

Solution :

Quand j'ai installé slapd, je n'étais pas connectée en tant que root. Or, c'était indispensable pour pouvoir exécuter slapd (qui se trouve dans le répertoire ldap-3.3/etc). Alors, mon maître de stage a changé mes droits sur slapd en exécutant les commandes suivantes en tant que root:

```
grincheux.crihan.fr# chown root slapd
grincheux.crihan.fr# chmod 4755 slapd
```

Ainsi, mes droits sur l'exécutable slapd sont :

```
grincheux.crihan.fr% ls -al ldap-3.3/etc/slapd
-rwsr-xr-x 1 root crihan 288748 Sep 8 14:15 ldap-3.3/etc/slapd
grincheux.crihan.fr%
```

Je peux ensuite exécuter slapd (afin de démarrer le serveur) comme si j'étais connectée en tant que root. Pour résoudre le problème que j'avais lors du "make install", il a fallu que mon maître de stage, Béatrice Gatto, supprime slapd en étant root car je ne pouvais pas le faire en simple utilisatrice.

❖ Problème à l'exécution de la commande "make test" :

Pour voir si notre serveur slapd fonctionne, il faut exécuter un "make test". Sur Linux, une erreur peut survenir au 7^{ème} test et le message est le suivant :

```
ns:~/ldap-3.3$ make test
....

>>>> Test succeeded
>>>> ./scripts/test006-acls completed OK.

>>> Starting ./scripts/test007-replication ...
Cleaning up in ./test-db...
Cleaning up in ./test-repl...
Starting master slapd on TCP/IP port 9009...
Starting slave slapd on TCP/IP port 9010...
Using ldapsearch to check that master slapd is running...
Using ldapsearch to check that slave slapd is running...
Starting slurpd...
Using ldapmodify to populate the database...
Using ldapmodify to modify the database...
Waiting 10 seconds for slurpd to send changes...
Using ldapsearch to read all the entries from the master...
Using ldapsearch to read all the entries from the slave...
kill: (20961) - No such pid
Comparing retrieved entries from master and slave...
./test-db/master.out ./test-db/slave.out differ: char 1, line 1
test failed - master and slave databases differ
>>>> ./scripts/test007-replication failed (exit 1)
make[1]: *** [all] Error 1
make[1]: Leaving directory `/home/bg/ldap-3.3/tests'
make: *** [test] Error 2
ns:~/ldap-3.3$
```

Sur cette machine ns.insa-rouen.fr, l'installation de LDAP se faisait dans le répertoire /home/bg/ldap-3.3.

Une autre personne a eu le même problème et l'a soumis à la liste LDAP (voir plus bas le paragraphe sur les listes). Une réponse a été envoyée et la voici :

```
Message-ID: <35781F2E.96F541B2@netscape.com>
Date: Fri, 05 Jun 1998 09:39:10 -0700
From: "Gordon Good" <ggood@netscape.com>
X-Mailer: Mozilla 4.09 [en] (WinNT; I)
X-Accept-Language: en
MIME-Version: 1.0
To: "Jon W. Backstrom" <vikings@iowa.net>
CC: ldap-support@umich.edu
Subject: Re: Specific test007 error...
References: <199806050016.TAA28113@virtual.iowa.net>
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
```

I suspect that the Linux port doesn't support threads. The replication daemon, slurpd, requires thread support. I realize that Linux probably has thread support now, but slurpd hasn't been updated since before thread support was available. If you don't care about using slurpd, you can just ignore the test failure. It's the last test; the rest of your code has checked out.

-Gordon

En fait, slurpd ne peut pas fonctionner sur Linux. Mais, pour notre annuaire, nous n'allons pas utiliser slurpd ; alors le 7^{ème} test n'a pas besoin de fonctionner correctement tant que les 6 autres marchent.

L'installation devient simple si on configure correctement le fichier Make-common. Cependant, si des problèmes se trouvent difficiles à résoudre, il est parfois utile de s'inscrire à une liste ldap comme ldap-support.

IV) La liste LDAP

Il est possible de s'inscrire à une liste concernant LDAP. Cette liste (ou mailing list en anglais) permet aux personnes travaillant sur LDAP de poser des questions à la liste ou de répondre à ces questions.

Je me suis inscrite à la liste ldap de l'Université du Michigan en envoyant un mail à l'adresse : ldap@umich.edu . Le mail doit contenir un seul mot : subscribe. Un mail de confirmation nous est retourné pour nous dire que l'on fait partie de la liste.

Ainsi, j'ai reçu dans ma boîte aux lettres des courriers de personnes demandant de l'aide pour des problèmes rencontrés en installant ou en utilisant LDAP. J'ai pu, moi aussi, envoyer un courrier pour un problème d'accents (voir chapitre sur les accents) en envoyant un mail à ldap@umich.edu.. Les personnes faisant partie de la liste reçoivent les même courrier et répondent (en écrivant à la liste pour que tout le monde ait la réponse) quand ils ont la solution.

Pour se retirer de la liste, il suffit d'envoyer un mail à ldap@umich.edu ou à la liste concernée. Ce mail doit contenir un seul mot : unsubscribe.

Les archives de cette liste sont disponibles sur le site : <http://www.umich.edu/~dirsvcs/ldap> et permettent d'avoir les réponses à des questions déjà posées.

Une fois l'installation (make et make install) faite correctement , il nous faut créer la base et apprendre à faire fonctionner le serveur slapd.

CHAPITRE VI : LE SERVEUR SLAPD

L'installation terminée et le serveur Slapd prêt à fonctionner, il nous reste à démarrer le serveur, créer une base et tester les différentes commandes disponibles. C'est ce que nous allons voir dans ce chapitre.

I) Démarrer et arrêter le serveur :

Pour démarrer le serveur, on utilise l'exécutable "*slapd*" qui se trouve dans le répertoire `ldap-3.3/etc` ou dans le répertoire `ETCDIR` (indiqué dans le fichier `Make-common`).

```
grincheux.crihan.fr% pwd
/usr/local/home/dfontain/ldap-3.3/etc
grincheux.crihan.fr% ls
base_ldif          ldapfilter.conf-    psslapd
base_ldif_uf       ldapfriendly         rcpt500
centipede          ldapfriendly-       rcpt500.help
fax500             ldapsearchprefs.conf rp500
fic.conf           ldapsearchprefs.conf- slapd
fichier.conf       ldaptemplates.conf  slapd.args
fichldif           ldaptemplates.conf- slapd.at.conf
ficldif_int        ldbm2ldif            slapd.at.conf-
ficldif_int2       ldbmcat              slapd.conf
genre_ldbm         ldbmtest             slapd.conf-
go500              ldif                 slapd.oc.conf
go500gw            ldif2id2children    slapd.oc.conf-
go500gw.help       ldif2id2entry       slapd.pid
go500gw.help-     ldif2index          slurpd
in.xfingerd        ldif2ldb             x
ldapfilter.conf    mail500
```

```
grincheux.crihan.fr%
```

Cette commande a plusieurs options :

- ♣ -f : cette option permet de spécifier le fichier de configuration utilisé pour démarrer le serveur (voir le paragraphe « fichier de configuration »). Le fichier utilisé par défaut est le fichier `ldap-3.3/etc/slapd.conf`.
- ♣ -p : cette option permet de spécifier le numéro de port sur lequel on démarrer le serveur. Le numéro par défaut est 389.
- ♣ -d : permet d'activer le mode debug. Si on indique cette option, le serveur inscrit des messages à l'écran quand le serveur slapd travaille. Sinon, le serveur travaille en tâche de fond.

Voyons la différence avec ou sans l'option -d :

Si on est en mode debug :

Notre fichier de configuration s'appelle fic.conf et on démarre le serveur sur le port 389 (port par défaut).

En mode debug, on *démarre le serveur* avec l'option -d en indiquant le niveau de débogage désiré, ici 1. Le niveau de débogage est un entier qui correspond à un certain type d'informations affichées.

```
grincheux.crihan.fr% slapd -d 1 -f fic.conf
slapd 3.3 (Tue Sep  8 15:36:28 MET DST 1998)
dfontain@grincheux.crihan.fr:/usr/local/home/dfontain/ldap-
3.3/servers/slapd
slapd starting
```

```
^Cgrincheux.crihan.fr%
```

Des messages apparaissent donc à l'écran. Pendant ce temps, nous n'avons plus la main sur cette fenêtre.

Exemple de messages affichés lors d'une recherche :

```
grincheux.crihan.fr% slapd -d 1 -f fichier.conf
slapd 3.3 (Tue Sep  8 15:36:28 MET DST 1998)
dfontain@grincheux.crihan.fr:/usr/local/home/dfontain/ldap-3.3/servers/slapd
slapd starting
do_bind

...

=> id2entry( 1 )
<= id2entry 0x64d28 (cache)
=> id2entry( 2 )
=> ldbm_cache_open( "/usr/local/home/dfontain/ldap-3.3/base/id2entry.gdbm", 2, 600 )
<= ldbm_cache_open (cache 1)
=> str2entry
<= str2entry 0x64d50
<= id2entry( 2 ) 0x64d50 (disk)
=> send_search_entry (cn=PRIGENT Herve, o=crihan, c=FR)
<= send_search_entry
send_ldap_result 0::
do_unbind
```

Pendant ce temps, le client affiche les enregistrements trouvés dans la fenêtre où la commande a été tapée :

```
grincheux.crihan.fr% ldapsearch -B -b "o=crihan, c=fr" 'cn=p*'
cn=PRIGENT Herve, o=crihan, c=FR
cn=PRIGENT Herve
sn=PRIGENT
givenname=Hervé
businesscategory=Responsable Service Informatique
mail=Herve.Prigent@crihan.fr
telephonenumber=+33 (0)2 35 59 61 59
facsimiletelephonenumber=+33 (0)2 35 59 61 40
objectclass=top
objectclass=person
objectclass=organizationalPerson
objectclass=inetOrgPerson
grincheux.crihan.fr%
```


Pour *arrêter le serveur* en mode debug, il suffit de taper "**^C**" (contrôle c). Et, nous avons de nouveau la main.

Si on n'est pas en mode debug :

Pour *démarrer le serveur*, on tape la commande :

```
grincheux.crihan.fr% slapd -f fic.conf
grincheux.crihan.fr%
```

La main nous est tout de suite rendue. Nous ne voyons donc pas si le serveur travaille ou non. Pour voir s'il est démarré ou non, on utilise la commande "ps -ef" qui permet de voir les processus en cours. Nous utilisons " | grep slapd" pour n'afficher que les lignes du résultat de la commande précédente concernant slapd.

```
grincheux.crihan.fr% ps -ef | grep slapd
dfontain 2032 1620 0 14:45:03 pts/5 0:00 grep slapd
grincheux.crihan.fr% slapd -f fic.conf
grincheux.crihan.fr% ps -ef | grep slapd
root 2034 1 1 14:45:10 ? 0:00 slapd -f fic.conf
dfontain 2036 1620 0 14:45:12 pts/5 0:00 grep slapd
grincheux.crihan.fr%
```

La première fois que l'on a utilisé la commande ps ici, le processus correspondant à la commande slapd -f fic.conf n'existe pas, alors le serveur ne « tourne » pas. On le démarre, et la deuxième fois le processus apparaît, il a le numéro 2034 comme numéro de processus.

Pour *arrêter un serveur* démarré sans le mode debug, il faut tuer le processus correspondant avec la commande "kill numéro_de_processus". Ici, ce sera :

```
grincheux.crihan.fr% kill 2034
grincheux.crihan.fr% ps -ef | grep slapd
dfontain 2038 1620 0 14:45:23 pts/5 0:00 grep slapd
grincheux.crihan.fr%
```

On exécute à nouveau la commande ps pour vérifier que le processus a été tué et que le serveur ne tourne plus. Quand on fait une recherche dans ce cas, il n'y a que le client qui affiche le ou les enregistrements trouvés.

II) Problèmes rencontrés lors du démarrage du serveur :

En fait, il n'y a qu'un seul problème rencontré qui ne soit pas lié à l'installation. Ce problème n'est visible que si on démarre le serveur en mode debug (option -d).

Le message d'erreur qui apparaît est le suivant :

```
grincheux.crihan.fr% slapd -d 1 -f fic.conf
slapd 3.3 (Tue Sep  8 15:36:28 MET DST 1998)
  dfontain@grincheux.crihan.fr:/usr/local/home/dfontain/ldap-
3.3/servers/slapd
bind() failed errno 125 (Address already in use)
grincheux.crihan.fr%
```

Cette erreur indique que le serveur tourne déjà sur le port indiqué avec l'option -p (389 sinon). Il suffit soit d'arrêter le serveur qui tourne pour le redémarrer soit de le démarrer sur un autre port.

Sans l'option -d, on ne remarque rien mais si on tape deux fois de suite la commande "slapd -f fic.conf", le serveur ne va démarrer que la première fois. La seconde fois, la commande est ignorée. Nous pouvons voir ça à l'aide de la commande ps :

```
grincheux.crihan.fr% slapd -f fic.conf           (on démarre le serveur une première fois)
grincheux.crihan.fr% ps -ef | grep slapd        (le processus correspondant est 2049)
  root   2049   1  0 14:46:30 ?                0:00 slapd -f fic.conf
dfontain 2053 1620 0 14:46:43 pts/5            0:00 grep slapd
grincheux.crihan.fr% slapd -f fic.conf           (on démarre le serveur une seconde fois)
grincheux.crihan.fr% ps -ef | grep slapd        (le processus qui tourne est toujours 2049)
  root   2049   1  0 14:46:30 ?                0:00 slapd -f fic.conf
dfontain 2057 1620 0 14:46:53 pts/5            0:00 grep slapd
grincheux.crihan.fr% slapd -d 1 -f fic.conf      (on démarre le serveur avec l'option -d cette fois)
slapd 3.3 (Tue Sep  8 15:36:28 MET DST 1998)
  dfontain@grincheux.crihan.fr:/usr/local/home/dfontain/ldap-
3.3/servers/slapd
bind() failed errno 125 (Address already in use)
grincheux.crihan.fr% ps -ef | grep slapd        (une erreur est apparue et le processus
correspondant à slapd est toujours 2049)
  root   2049   1  0 14:46:30 ?                0:00 slapd -f fic.conf
dfontain 2060 1620 0 14:47:05 pts/5            0:00 grep slapd
grincheux.crihan.fr% kill 2049                 (on tue ce processus 2049)
grincheux.crihan.fr% ps -ef | grep slapd        (il ne reste plus de serveur en fonctionnement)
dfontain 2062 1620 0 14:47:15 pts/5            0:00 grep slapd
grincheux.crihan.fr%
```

C'est toujours le premier processus qui tourne, jusqu'à ce qu'on le tue.

III) Fichier de configuration :

Par défaut, le fichier de configuration pour démarrer slapd est le fichier slapd.conf situé dans le répertoire ETCDIR indiqué dans le Make-common. Ce fichier contient :

```
include          /usr/local/home/dfontain/ldap-3.3/etc/slapd.at.conf
include          /usr/local/home/dfontain/ldap-3.3/etc/slapd.oc.conf
schemacheck     off
referral        ldap://ldap.itd.umich.edu

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

database        ldbm
suffix          "o=Your Organization Name, c=US"
directory       /usr/tmp
rootdn          "cn=root, o=Your Organization Name, c=US"
rootpw          secret
```

Il faut remplacer "Your Organization Name, c=US" et "cn=root, o=Your Organization Name, c=US" par ce qui correspond à notre base. La ligne contenant le répertoire /usr/tmp doit aussi être changée. Voyons à quoi correspondent les différentes lignes de ce fichier et comment écrire notre propre fichier de configuration :

Les lignes blanches et les lignes de commentaires commençant par # sont ignorées. Les lignes commençant par un espace sont considérées comme la suite de la ligne précédente.

Nous allons voir certaines options, c'est-à-dire celles que nous avons utilisées, mais pas toutes car il y en a beaucoup. Si vous voulez les connaître toutes, elles sont bien détaillées dans le guide "The SLAPD and SLURPD Administrator's Guide" de l'Université du Michigan.

Notre fichier de configuration se nomme fic.conf et contient :

```
referral        ldap://ldap.itd.umich.edu
database        ldbm
suffix          "o=crihan, c=fr"
directory       /usr/local/home/dfontain/ldap-3.3/base
rootdn          "cn=Delphine Fontaine, o=crihan, c=fr"
rootpw          s
```

Options utilisées (générales à toutes les bases de données) :

- `referral <url>` : Cette option indique l'endroit où la recherche se fera si aucune base locale n'a été trouvée.
- `database <type de base de données>` : cette option (indispensable) indique le type de base de données utilisée : `ldbm`, `shell` ou `passwd` selon ce qui a été spécifié dans le `Make-common`.

Exemple : `database ldbm`

Ceci marque le début de la définition d'une base de données de type `ldbm`.

- `rootdn <dn>` : cette option permet d'indiquer le DN (distinguished name) qui n'est pas sujet aux contrôles d'accès sur la base de données.

Exemple : `rootdn "cn=Fontaine Delphine, o=crihan, c=fr"`

- `rootpw <mot de passe>` : Cette option permet d'indiquer le mot de passe pour le `rootdn` indiqué ci-dessus.

Exemple : `rootpw secret` (le mot de passe est donc `secret`)

- `suffix <dn suffix>` : Cette option (indispensable) indique le suffixe du dn qui va appartenir à cette base. Il faut en indiquer au moins un mais on peut en mettre plusieurs.

Exemple : `suffix "o=crihan, c=fr"`

`suffix "o=insa, c=fr"`

Les recherches se feront sur les enregistrements dont le DN se termine par `"o=crihan, c=fr"` ou par `"o=insa, c=fr"`.

Options utilisées (spécifiques au type LDBM) :

C'est en effet la seule base de données finale (backend database) qui a été utilisée pendant ce stage.

- `directory <répertoire>` : Cette option (indispensable) permet d'indiquer le répertoire où se trouvent les fichiers contenant la base de données, ce sont les fichiers `id2entry.gdbm`, ... pour nous.

Exemple : `directory /usr/local/home/dfontain/ldap-3.3/base`

Les fichiers seront mis dans le répertoire `/usr/local/home/dfontain/ldap-3.3/base`.

- `index {<attrlist> | default} [pres,eq,approx,sub,none]` :

Cette option n'est pas obligatoire. Il existe quatre types d'index : `pres`, `eq`, `approx`, et `sub`.

- ◆ `pres` : L'index `pres` (*presence*) est le plus simple des index. Il permet de faire des recherches telles que : `telephonenumber=*` ; ce qui permet de rechercher tous les enregistrements ayant un numéro de téléphone.
- ◆ `eq` : L'index `eq` (*equality*) permet de rechercher des enregistrements contenant un attribut ou champ spécifique comme par exemple : `cn=Delphine Fontaine`.

- ◆ approx : L'index approx (approximative) permet des recherches approximatives. Par exemple, si un enregistrement contient `cn=Robert E Lee`. Une recherche approximative va donner une réponse pour les requêtes suivantes :

```
cn~=Robert Lee,  
cn~=Robert  
cn~=Lee.
```

- ◆ sub : L'index sub (*substring*) permet des recherches comme `cn=*derson`. Cette recherche va afficher tous les enregistrements finissant par `derson` comme

```
Bill Anderson  
Jill Anderson  
Steve Sanderson
```

De même, la recherche `telephonenumber= *555*` va donner tous les enregistrements dont le numéro de téléphone contient 555.

CHAPITRE VII : **CREATION D'UNE** **BASE DE DONNEES**

Dans les chapitres VII et VIII, des exemples accentués seront utilisés mais le problème des accents sera développé dans le chapitre IX.

I) Le format LDIF :

Avant de créer notre base de données (afin d'effectuer des opérations telles qu'une recherche, l'addition d'un enregistrement, etc.), il faut entrer un minimum d'enregistrements dans un fichier texte. Les enregistrements seront au format LDIF (LDAP Data Interchange Format).

Un enregistrement (ou entrée) doit suivre le schéma suivant :

```
[<id>]
dn : <distinguished name>
objectClass : <classe d'objet>
objectClass : <classe d'objet>
.....
<type d'attribut> :<valeur d'attribut>
<type d'attribut> :<valeur d'attribut>
.....
```

Chaque enregistrement doit avoir :

- un attribut nommé DN qui permet d'identifier l'enregistrement.
- au moins un nom de classe d'objet
- un attribut faisant partie de la classe objet spécifiée juste avant.

Les autres attributs et classes objets sont optionnels.

Voyons quels sont les différents éléments de cet enregistrement :

- [*<id>*] : c'est le numéro de l'enregistrement. Quand on génère la base, ce numéro est généré automatiquement. Il ne faut pas ajouter ou modifier cette valeur nous-mêmes.
- dn : *<distinguished name>* : indique le dn (ou champs distinctif) de l'enregistrement
- objectclass :*<classe d'objet>* : spécifie le type de classe d'objet et donc le type d'attributs que l'on peut utiliser.
- *<type d'attribut>* : spécifie le type d'attribut que l'on utilise
- *<valeur de l'attribut>* : spécifie la valeur de cet attribut

Le champ "dn" :

Le champs dn (ou champs distinctif), comme nous l'avons déjà vu, permet d'identifier un enregistrement pour les recherches par exemple.

Le format de ce champs est le suivant :

cn=nom d'usage, [street=adresse, l=lieu, st = état ou province, ou=service, o=organisation], c=nom du pays

Chaque élément du dn doit être séparé par une virgule qui peut être suivie d'un espace.

Les attributs du dn doivent toujours être dans le même ordre puisque il représente un chemin qui doit être en accord avec la hiérarchie en arbre. Par exemple, les deux champs distinctifs ci-dessous ne représentent pas le même enregistrement :

dn: cn=Marie Dupont, ou=secrétariat, o=entreprise X, c=FR

dn: cn=Marie Dupont, o=entreprise X, ou=secrétariat, c=FR

Souvent le dn commence par le champ "common name" (cn) mais si l'annuaire contient l'enregistrement défini par :

dn: cn=Marie Dupont, ou=secrétariat, o=entreprise X, c=FR

alors il doit avoir les enregistrements commençant par :

dn: ou=secrétariat, o=entreprise X, c=FR

et

dn: o=entreprise X, c=FR

Ces deux enregistrements doivent être situés avant les enregistrements dont le champ dn contient "o=entreprise X, c=FR" ou "ou=secrétariat, o=entreprise X, c=FR".

Les différents éléments du champ dn :

- **c** (country ou pays) : Pays où se situe la personne ou l'objet défini par l'enregistrement. Il est composé de deux lettres. Exemple : c=US, c=FR, c=GB ...
- **cn** (common name ou nom d'usage) : Cet attribut définit l'enregistrement.
- **l** (locality ou lieu) : Cet attribut peut être une ville, une région, ...
- **o** (organization ou organisme) : Exemple :o=INSA, o=CRIHAN ...
- **ou** (organizational unit ou service) : C'est le service ou le département dont fait partie la personne ou l'objet défini par l'enregistrement.
- **st** (state ou état) : C'est le nom de l'état ou de la province où se situe la personne ou l'objet défini par l'enregistrement
- **street** : c'est l'adresse où réside l'objet ou la personne défini par l'enregistrement.

Pour le CRIHAN, les champs dn ne contiennent que les attributs cn, o et c. Le format est donc :

dn: cn=NOM Prénom, o=CRIHAN, c=FR

Voici quelques exemples :

dn: o=crihan, c=FR

dn: cn=CHARTON Beatrice, o=crihan, c=FR

dn: cn=DOMALAIN Chrystel, o=crihan, c=FR

dn: cn=MASSIOT Alain, o=crihan, c=FR

Les champs "objectclass" :

Le type de l'enregistrement est défini par les valeurs des attributs nommés objectclass. En fait, on indique que l'enregistrement définit une personne appartenant à une organisation grâce à ses attributs.

Ainsi, chaque objectclass permet d'utiliser certains attributs ou champs. Par exemple, un enregistrement dont le champ objectclass a la valeur "organizational person" peut avoir comme attributs le nom, le prénom, l'adresse email,

Remarque : Un enregistrement peut avoir un ou plusieurs champs "objectClass".

Dans l'annuaire du Crihan, et dans celui de l'INSA, nous allons écrire des enregistrements définissant des personnes appartenant à une organisation. L'enregistrement sera donc du type :

```
dn: distinguished name
cn: NOM Prénom
sn: NOM
liste des champs optionnels
...
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

Voyons ce qu'indiquent ces différentes lignes :

dn: *distinguished name*

c'est le champ distinctif qui définit l'enregistrement. Exemple :

```
dn : cn=FONTAINE Delphine, o=CRIHAN, c=FR
```

cn: *NOM Prénom*

c'est le nom d'usage de la personne, c'est-à-dire son Nom de famille et son prénom. Exemple :

```
cn : FONTAINE Delphine
```

sn: *NOM*

c'est le Nom de la personne. Exemple :

```
sn : FONTAINE
```

liste des champs

Ce sont les différents attributs que l'on peut ajouter comme le prénom, l'adresse email, le téléphone, etc...

objectClass: top

Spécifie le début de la déclaration des différents types de classe d'objet. Cette ligne est optionnelle mais d'anciens clients LDAP en ont besoin pour leur recherches.

objectClass: person

On précise le type de classe d'objet personne pour indiquer que l'enregistrement définit une personne.

objectClass: organizationalPerson

On précise que la personne définie dans cet enregistrement fait partie d'une organisation.

objectClass: inetOrgPerson

Ce type de classe d'objet est recommandé pour les personnes faisant partie d'une organisation car elle permet d'utiliser une grande variété d'attributs.

Exemple de fichier LDIF :

Cet exemple montre un fichier LDIF contenant trois personnes faisant partie du CRIHAN :

```
dn: o=crihan, c=FR
o: crihan
objectclass: organization

dn: cn=CHARTON Beatrice, o=crihan, c=FR
cn: CHARTON Beatrice
sn: CHARTON
givenname: Béatrice
mail: Beatrice.Charton@crihan.fr
businesscategory: Ingénieur Système
telephoneNumber: +33 (0)2 35 59 61 59
facsimileTelephoneNumber: +33 (0)2 35 59 61 40
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson

dn: cn=PRIGENT Herve, o=crihan, c=FR
cn: PRIGENT Herve
sn: PRIGENT
givenname: Hervé
businesscategory: Responsable Service Informatique
mail: Herve.Prigent@crihan.fr
telephoneNumber: +33 (0)2 35 59 61 59
facsimileTelephoneNumber: +33 (0)2 35 59 61 40
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson

dn: cn=VANDROMME Dany, o=crihan, c=FR
cn: VANDROMME Dany
sn: VANDROMME
givenname: Dany
mail: Dany.Vandromme@crihan.fr
businesscategory: Président
telephoneNumber: +33 (0)2 35 59 61 59
facsimileTelephoneNumber: +33 (0)2 35 59 61 40
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
```

Les différents attributs :

Voici les attributs utilisés pour le CRIHAN mais il en existe beaucoup d'autres

Nom de l'attribut	Description
businessCategory	Cet attribut indique la profession de la personne définie par l'enregistrement. Par exemple : <code>businessCategory : Ingénieur</code>
mail	Adresse email. Par exemple : <code>mail : dfontain@crihan.fr</code>
facsimileTelephoneNumber (fax)	Numéro de fax. Par exemple : <code>facSimileTelephoneNumber : 02-35-00-01-02</code>
givenName	Prénom. Par exemple : <code>givenName : Delphine</code>
commonName (cn)	Nom d'usage. Par exemple : <code>cn : FONTAINE Delphine</code>
surname (sn)	Nom de famille. Par exemple : <code>sn : FONTAINE</code>
localityName (l)	Lieu (la ville en général). Par exemple : <code>l : Mont Saint Aignan</code>
organization Name (o)	Le nom de l'organisation. Par exemple : <code>o : CRIHAN</code>
telephoneNumber	Numéro de téléphone. Par exemple : <code>telephoneNumber : 02-35-11-22-33</code>
organizationalUnitName (ou)	Nom d'un service. Par exemple : <code>ou : secrétariat 1er cycle</code>

II) Création d'une base de données avec la commande ldif2ldb :

Une fois le fichier au format LDIF terminé, nous pouvons générer la base grâce à la commande `ldif2ldb` (dont l'exécutable se trouve dans le répertoire `ETCDIR` indiqué dans le fichier `Make-common`). La base sera créée dans le répertoire spécifié dans le fichier `slapd.conf`. Pour nous, c'est dans le répertoire `/usr/local/home/dfontain/ldap-3.3/base`.

La commande `ldif2ldb` permet de générer une base à partir d'un fichier LDIF. Sa syntaxe est :

```
ldif2ldb -i ldif_file -f slapd.conf
```

où `ldif_file` est le fichier contenant les enregistrements au format LDIF et `slapd.conf` le fichier de configuration du serveur slapd.

Pour nous, `ldif_file` est remplacé par `base_ldif_utf` qui contient la base au format utf (cf. chapitre concernant les accents) et `slapd.conf` sera remplacé par `fic.conf` (cf. chapitre précédent).

Remarque : Cette commande va écraser l'ancienne base pour en générer une nouvelle à partir des fichiers indiqués.

Les options pour cette commandes sont :

-i

Ce paramètre est obligatoire et indique le fichier contenant les enregistrements au format LDIF.

-f

Ce paramètre indique le fichier de configuration et n'est pas obligatoire. Par défaut, c'est le fichier slapd.conf qui est utilisé.

-d

Indique le niveau de débogage que l'on désire comme pour la commande slapd (cf. chapitre précédent)

-v

Indique que l'on veut utiliser le mode verbeux.

Exemple d'utilisation de cette commande :

```
grincheux.crihan.fr% ldif2ldb -i base_ldif_utf -f fic.conf
```

III) Utilisation de la commande ldbmcat :

On peut faire l'inverse, c'est-à-dire générer un fichier LDIF à partir de la base de données à l'aide de la commande ldbmcat.

Remarque : Avant d'utiliser la commande ldbmcat (dont l'exécutable se trouve dans le répertoire ETCDIR), il faut s'assurer que le serveur ne tourne pas.

On peut ensuite taper la commande suivante :

```
ldbmcat -n fichier_id2entry > fichier_de_sortie
```

où fichier id2entry est le nom du fichier contenant la base et fichier_de_sortie le fichier où sera écrit le résultat de la commande. L'option -n permet de ne pas afficher les numéros des enregistrements.

Pour nous, le fichier contenant la base est /usr/local/home/dfontain/ldap-3.3/base/id2entry.gdbm

La commande sera donc :

```
ldbmcat -n id2entry.gdbm > base_ldif
```

Maintenant que l'on sait créer une base et démarrer un serveur, voyons quelles sont les différentes commandes disponibles.

CHAPITRE VIII : **LES COMMANDES LDAP**

Quand le serveur LDAP a été installé, un client a été installé en même temps. Ce client permet à un utilisateur de rechercher, modifier, effacer, ... des enregistrements dans l'annuaire.

Les commandes disponibles sont les suivantes :

- ldapsearch
- ldapadd
- ldapdelete
- ldapmodify
- ldapmodrdn

Toutes ces commandes se trouvent dans le répertoire `/usr/local/home/dfontain/ldap-3.3/client/tools`.

I) Utilisation du format LDIF :

Pour ajouter ou modifier un enregistrement dans un annuaire, il faut utiliser le format LDIF. En effet, c'est le même format que celui décrit dans le chapitre précédent pour remplir le fichier contenant tous les enregistrements.

Les commandes `ldapadd`, `ldapmodify`, et `ldapmodrdn` nous permettent de rentrer des données soit par un fichier (avec l'option `-f`) soit par la ligne de commande.

Si on n'utilise pas l'option `-f`, chaque ligne est entrée comme si elle était lue dans un fichier. Quand la saisie est terminée, il faut entrer le caractère que le shell utilisé reconnaît comme étant la fin d'un fichier (EOF). Le plus souvent, c'est le caractère `^D` (control-D). Nous allons voir plusieurs exemples pour chaque commande.

Pour toutes les commandes, quand on utilise un attribut avec des espaces, asterix [`*`], backslash [`\`] ou autre, il faut les mettre entre guillemets. Exemple :

```
-D "cn=Delphine Fontaine, o=crihan, c=FR"
```

De plus, la plupart des commandes ont un accès limité. En effet, pour pouvoir utiliser ces commandes, il faut indiquer le nom distinctif (DN) de la personne ayant le droit de faire des modifications dans la base ; c'est le champ `"rootdn"`. Un mot de passe pour cette personne est nécessaire ; c'est le champ `"rootpw"`. (cf. chapitre « Le serveur Slapd »).

II) La commande ldapsearch :

Cette commande permet de rechercher un enregistrement dans un annuaire existant avec certains critères de sélection.

La syntaxe est la suivante :

```
ldapsearch [-h <machine_hôte>] [-p <port>] [-D <rootdn>] [-w  
<mot_de_passe>] -b <base> [-s <scope>] [-f <fichier> | <filtre >] [<champs>] [-S  
<champ>] [-a <deref>] [-F <sep>] [-O <maxhops>] [-Z -P <certdbpath>] [-ABHnoRtuv]
```

Les options de ldapsearch sont nombreuses. Voyons celles qui sont les plus utilisées :

-D

Indique le champ dn de celui qui a le droit de rechercher.

-w

Indique le mot de passe pour la valeur du champ dn donné précédemment.

-h

Indique le nom de la machine où est installé le serveur slapd.

-p

Indique le numéro de port sur lequel est le serveur. Ce numéro est 389 par défaut. Si l'option **-Z** est utilisée, le port par défaut est 636..

-b

Indique le point de départ de la recherche. Cette valeur doit être un champ dn qui existe dans la base de données. Ce paramètre n'est pas nécessaire si la variable LDAP_BASEDN a été définie dans le fichier ldapconfig.edit.h. Cette valeur doit être mise entre guillemets. Exemple :

```
-b "cn=Delphine Fontaine, o=crihan, c=FR"
```

ou

```
-b " o=crihan, c=FR"
```

-f

Indique le fichier où se trouvent les filtres à utiliser pour la recherche.

-d

Indique le niveau de débogage désiré..

-S

Indique les critères pour trier les enregistrements trouvés. Par défaut, les enregistrements ne sont pas triés. Si on met **-S ""**, les enregistrements sont triés selon leur champ dn.

-B

Affiche les caractères non-ascii (surtout utilisée pour afficher les accents)

La syntaxe pour une *simple recherche* est :

```
attribut opérateur valeur
```

Par exemple :

```
cn = Fontaine Delphine
```

cn est l'attribut, = l'opérateur, et "Fontaine Delphine" la valeur.

Les différents opérateurs sont :

=	retourne l'enregistrement pour lequel on a la bonne valeur pour le champ indiqué <i>exemple</i> : cn=Fontaine Delphine retourne l'enregistrement pour lequel le champ cn a pour valeur Fontaine Delphine.
=<chaîne_de_caractère>	retourne les enregistrements contenant la chaîne de caractère spécifiée <i>exemple</i> : cn=*Delp* cn=Font* cn=Font*elphine
>= (supérieur ou égal)	<i>exemple</i> : employeenumber >= 100 retourne les enregistrements contenant l'attribut indiqué (ici employeenumber) et dont la valeur est supérieure ou égale à la valeur indiquée
<=	c'est le même principe que >= <i>exemple</i> : cn<=fontaine recherche tous les enregistrements dont le champs cn est entre "cn=a" et "cn=fontaine".
=*	retourne tous les enregistrements contenant l'attribut indiqué <i>exemple</i> : cn=*
~=	retourne les enregistrements contenant l'attribut indiqué et qui est à peu près égal à la valeur indiquée <i>exemple</i> : cn~=fontaine

Exemple de recherches simples :

Supposons que :

- Nous voulons faire une recherche avec les filtres se trouvant dans le fichier searchdb.
- Nous voulons faire les recherches à partir du sous-arbre dont la racine est "o=crihan, c=FR".
- Le serveur est sur la machine grincheux.
- Le serveur utilise le port 654.

Il faut entrer les filtres dans le fichier searchdb et la commande suivante peut être tapée :

```
ldapsearch -b "o=crihan, c=FR" -h grincheux -p 654 -f searchdb
```

Si on ne met pas les filtres de recherche dans un fichier, on peut taper la commande suivante pour trouver tous les enregistrements ayant la ligne "objectclass=person" :

```
ldapsearch -b "o=crihan, c=FR" -h grincheux -p 654 "objectclass=person"
```

Supposons maintenant que :

- Nous voulons rechercher tous les enregistrements dont le champ sn (cf. chapitre précédent pour les différents champs utilisés) contient la chaîne de caractère "peg".
- Le serveur est sur la machine grincheux, sur le port 389.
- Nous voulons faire les recherches à partir du sous-arbre dont la racine est "o=crihan, c=FR".

La commande sera la suivante :

```
ldapsearch -h grincheux -b "o=insa,c=fr" 'sn=*peg*'
```

Si on veut, dans les mêmes conditions, afficher tous les enregistrements contenant un numéro de téléphone, on peut exécuter la commande :

```
ldapsearch -h grincheux -b "o=insa,c=fr" 'telephonenumber=*'
```

Nous pouvons faire une recherche plus précise grâce aux filtres multiples. Ainsi, plusieurs critères peuvent être combinés en utilisant la syntaxe suivante :

```
(opérateur_logique((opération_de_recherche)(opération_de_recherche)(opération_de_recherche) ..))
```

ou

```
(opérateur_logique(opération_de_recherche)((opérateur(opération_de_recherche)(opération_de_recherche)) ..))
```

Où les opérations de recherches sont du même format qu'une recherche simple et où les différents opérateurs logiques sont :

Opérateur	Symbole	Description
Et	&	Tous les filtres doivent être vrais. Par exemple : (&(filtre)(filtre)(filtre)...)
Ou		Au moins un des filtres doit être vérifié. Par exemple : ((filtre)(filtre)(filtre)...)
Négation	!	Le filtre indiqué doit être faux. Par exemple : (!(cn=F*)) recherche les enregistrements dont le champs cn ne commence pas par la lettre f.

Exemples de filtres pour les recherches plus précises :

- (&(objectclass=person)(!(telephonenumber=*))

Ce filtre recherche tous les enregistrements ayant le champ objectclass=person et n'ayant pas de numéro de téléphone.

- (&(telephonenumber=*55*)(|(cn=Fontaine Delphine)(cn=Vandromme Dany))

Ce filtre recherche tous les enregistrements ayant un numéro de téléphone contenant 55 et dont la valeur du champ cn est soit "Fontaine Delphine" soit "Vandromme Dany".

Exemple de recherches plus compliquées :

- `grincheux.crihan.fr% ldapsearch -p 389 -h grincheux -B -b "o=insa, c=fr" '(&(cn=*drau*)(objectclass=top))'`

Cette recherche permet d'afficher tous les enregistrement qui contient la ligne `objectclass=top` et dont le champ `cn` contient la chaîne de caractère "drau".

- `grincheux.crihan.fr% ldapsearch -p 389 -h grincheux -B -b "o=insa, c=fr" '(&(cn=*drau*)(objectclass=top)(telephonenumber=*))'`

Cette recherche permet d'afficher tous les enregistrement qui contient la ligne `objectclass=top`, qui a un numéro de téléphone et dont le champ `cn` contient la chaîne de caractère "drau".

- `atchoum.crihan.fr% ldapsearch -p 389 -h grincheux -B -b "o=insa, c=fr" '(&(cn=boncrey didier)(telephonenumber=*85*))'`

Cette recherche permet d'afficher tous les enregistrement dont le champ `cn` est `cn=boncrey didier` et dont le numéro de téléphone contient 85. En sortie, nous avons :

```
cn=BONCREY Didier, o=crihan, c=fr
cn=BONCREY Didier
sn=BONCREY
givenname=Didier
mail=Didier.Boncrey@insa-rouen.fr
telephonenumber=8574
ou=insa
objectclass=top
objectclass=person
objectclass=organizationalPerson
objectclass=inetOrgPerson
```

Ceci correspond bien à la recherche.

III) La commande ldapadd :

Cette commande permet d'ajouter un enregistrement à un annuaire déjà existant.

Remarque : On ne peut ajouter un enregistrement si les étages supérieurs sont déjà dans l'annuaire. Par exemple, on peut ajouter l'enregistrement dont le dn sera : "`cn=FONTAINE Delphine, o=crihan, c=FR`" que si l'enregistrement dont le dn est "`o=crihan, c=FR`" existe déjà.

Chaque nouvel enregistrement que l'on ajoute doit avoir un champ dn identifiant l'enregistrement, au moins un nom de classe d'objet et un champ compatible avec la classe d'objet précisée.

La syntaxe de cette commande est :

```
ldapadd [-h <host>] [-p <port>] [-D <binddn>] [-w <password>] [-f <ldiffile>] [-e <rejectedfile>] [-O <maxhops>] [-Z -P <certdbpath>] [-bcFHnRv]
```


Voici les principales options de cette commande :

-f

Indique le fichier contenant le ou les enregistrements à ajouter. Si on ne met pas ce paramètre, on ajoute l'enregistrement directement sur la ligne de commande.

-D

Indique le champ distinctif (dn) de celui qui a le droit de modifier la base de donnée.

-w

Indique le mot de passe associé au dn indiqué ci-dessus.

-h

Indique le nom de la machine sur laquelle est installé le serveur.

-p

Indique le numéro de port qui est 389 par défaut. Si l'option **-Z** est utilisée, le numéro de port est 636.

Remarque : Les lignes entrées directement sur la ligne de commande doivent être au format LDIF (cf. chapitre précédent).

Exemples :

Voici un exemple d'ajout de deux enregistrement sans passer par un fichier (on est sur la machine atchoum qui possède un client ldap) :

```
grincheux% ldapadd -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h
grincheux
dn: cn=Toto, o=crihan, c=fr
cn: Toto
sn: Toto
objectClass: person
adding new entry cn=Toto, o=crihan, c=fr
^D

dn: cn=Titi, o=crihan, c=fr
cn: Titi
sn: Titi
telephonenumber: 02 35 36 11 22
objectClass: person
adding new entry cn=Titi, o=crihan, c=fr
^D

grincheux%
```

Une fois la première ligne tapée, nous sommes connectés au serveur qui est sur la machine grincheux (indiquée par l'option **-h**) sur le port 389 (indiqué par l'option **-p**). Ensuite, on tape les quatre lignes suivantes pour ajouter l'enregistrement dont le champ "dn" sera "cn=Toto, o=crihan, c=fr". On termine la saisie par ^D (control-D). La main ne nous est pas rendue, ce qui nous permet d'ajouter un deuxième enregistrement. La main nous est rendue seulement quand on tape "return" après le deuxième control-D.

Voici maintenant un exemple d'exécution de la commande ldap avec l'utilisation de l'option -f :

L'enregistrement à ajouter est mis dans le fichier newentry.ldif. Ce fichier contient :

```
dn: cn=Dupont Marie, o=crihan, c=fr
cn: Dupont Marie
sn: Dupont
objectClass: person
```

La commande sera donc :

```
atchoum% ldapadd -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h
grincheux -f newentry.ldif
adding new entry cn=Dupont Marie, o=crihan, c=fr

atchoum%
```

La main nous est rendue aussitôt.

Nous pouvons vérifier nos commandes à l'aide de la commande ldapsearch vu ci-dessus.

IV) La commande ldapdelete :

Cette commande permet de supprimer des enregistrements d'un annuaire existant. On ne peut pas supprimer des enregistrements ayant un ou plusieurs fils. Par exemple, on peut supprimer "cn=Fontaine Delphine, o=crihan, c=FR" mais pas "o=crihan, c=FR". Cependant, si l'enregistrement dont le champ "dn" est "o=crihan, c=FR" n'a aucun fils, il peut être enlevé.

La syntaxe de cette commande est :

```
ldapdelete [-h <host>] [-p <port>] [-D <binddn>] [-w <password>] [-f
<file_with_DNs>][-O <maxhops>] [-Z -P <certdbpath>] [-cHnRv] [<dn_to_delete>]
```

Pour utiliser cette commande, c'est le même principe que ldapadd. Les options sont les mêmes.

Exemple :

Voici un exemple sans utilisation de fichier :

```
grincheux% ldapdelete -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h
grincheux "cn=essai, o=crihan, c=fr"
grincheux%
```

On donne directement le champ "dn" de l'enregistrement que l'on veut supprimer. Ainsi, l'enregistrement défini par le champ "dn" indiqué (ici, "cn=essai, o=crihan, c=fr") est supprimé.

Voici un exemple avec l'utilisation du fichier fichdelete.ldif :

Le fichier contient la ligne :

```
cn=essai2, o=crihan, c=fr
```

La commande sera donc :

```
grincheux% ldapdelete -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h  
grincheux -f fichdelete.ldif  
grincheux%
```

V) La commande ldapmodrdn :

Cette commande permet de modifier le champ "RDN" d'un enregistrement existant. En fait, quand on change le RDN, on renomme l'enregistrement. Le RDN est la partie la plus à gauche du champ DN :

Exemples :

Le RDN pour :

```
cn=Fontaine Delphine, o=crihan, c=FR
```

est :

```
cn=Fontaine Delphine
```

Et le RDN pour :

```
o=crihan, c=FR
```

est :

```
o=crihan
```

Remarque :

On ne peut pas renommer un enregistrement qui a un ou plusieurs fils. Par exemple, on peut renommer `cn=Fontaine Delphine, o=crihan, c=FR` mais pas `o=crihan, c=FR`. Cependant, si l'enregistrement dont le champ dn est `o=crihan, c=FR` n'a aucun fils, il peut être renommé.

La syntaxe de cette commande est :

```
ldapmodrdn [-h <host>] [-p <port>] [-D <binddn>] [-w <password>] [-f <filename>] [-e <rejectedfile>] [-O <maxhops>] [-Z -P <certdbpath>] [-cFHnrRv] [dn rdn]
```

De même que pour les commandes précédentes, il faut indiquer un DN et un mot de passe pour pouvoir renommer un enregistrement.

Pour utiliser cette commande, les options sont les mêmes que pour les commandes précédentes.

Exemple :

Voici un exemple sans utilisation de fichier :

```
grincheux% ldapmodrdn -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h
grincheux
cn=Titi, o=crihan, c=fr
cn=Titi2
^D
grincheux%
```

On vérifie en faisant une recherche :

```
grincheux% ldapsearch -h grincheux 'cn=titi*'
cn=Titi2,o=crihan,c=fr
cn=Titi
objectclass=person
grincheux%
```

Voici un exemple avec l'utilisation du fichier `modif.ldif` :

Le fichier contient :

```
cn=Titi2, o=crihan, c=fr
cn=Titi3
```

La commande sera donc :

```
grincheux% ldapmodrdn -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h
grincheux -f modif.ldif
grincheux%
```

VI) Commande `ldapmodify` :

Cette commande permet de modifier un enregistrement en ajoutant, supprimant ou remplaçant un champ.

De même qu'avec les commandes précédentes, un DN et un mot de passe sont nécessaires.

La syntaxe de cette commande est :

```
ldapmodify [-h <host>] [-p <port>] [-D <binddn>] [-w <password>] [-f <ldiffile>] [-e <rejectedfile>] [-O <maxhops>] [-Z -P <certdbpath>] [-bcFHnRv]
```

Les options sont les mêmes que pour les commandes précédentes.

Format général :

- On indique le champ dn de l'enregistrement qui va être modifié.
- On indique le type de changement que l'on veut faire (add, delete, modify, ou modrdn).
- On indique les attributs à changer et leurs nouvelles valeurs.

Ainsi, on a le schéma suivant :

```
dn: nom distinctif
changetype: type de changement

opération: champ à modifier
champ: nouvelle valeur
...
-
opération: champ à modifier
champ: nouvelle valeur
...
-
...
```

Le tiret (-) est obligatoire car il permet de séparer chaque opération de changement. Si il n'y a qu'une seule opération à effectuer, il devient inutile. Par exemple :

```
grincheux.crihan.fr% ldapmodify -D "cn=Delphine Fontaine, o=insa,
c=fr" -w s -p 389 -h grincheux
dn: cn=PEGARD Francis, o=insa, c=fr
changetype:modify
add:ou
ou: Atelier de Mécanique
-
add:l
l: Madrillet
modifying entry cn=PEGARD Francis, o=insa, c=fr
^D

grincheux.crihan.fr%
```

Ici, on ajoute un champ "ou" (service) dont la valeur est "Atelier de Mécanique" et un champ "l" (lieu) dont la valeur est "Madrillet". Ces changements sont faits pour l'enregistrement dont le champ "dn" est "cn=PEGARD Francis, o=insa, c=fr". Le tiret nous permet de faire deux opérations (qui sont ici deux ajouts de champ) sur le **même** enregistrement.

Les différents types de changement :

La commande `ldapmodify` permet de faire tous les changements possibles puisque le type de changement (indiqué par la ligne `changetype: type de changement`) peut prendre quatre valeurs :

- **changetype: add**

Ce type de changement est utilisé pour ajouter un nouvel enregistrement. Le format de ce type de changement est :

```
dn: champ distinctif
changetype: add
objectClass: nom de la classe d'objet
objectClass: nom de la classe d'objet
...
type de champ: valeur du champ
type de champ: valeur du champ
...
```

- **changetype: delete**

Ce type de changement est utilisé pour supprimer un enregistrement existant. Le format de ce type de changement est :

```
dn: champ distinctif
changetype: delete
```

- **changetype: modrdn**

Ce type de changement est utilisé pour renommer un enregistrement existant. Le format de ce type de changement est :

```
dn: champ distinctif
changetype: modrdn
newrdn: nouveau rdn
deleteoldrdn: 0 ou 1
```

Le champ RDN (relative distinguished name) est le nom distinctif relatif de l'enregistrement comme on l'a vu dans le paragraphe précédent.

Si `deleteoldrdn` vaut 0, les valeurs des champs de l'ancien RDN sont insérés dans l'enregistrement correspondant au nouveau RDN. Si `deleteoldrdn` vaut 1, l'ancien RDN sera supprimé.

- **changetype: modify**

Ce type de changement est utilisé pour modifier un enregistrement existant. Il y a plusieurs changements possibles sur un enregistrement : on peut ajouter, supprimer ou modifier un champ de cet enregistrement.

Voici le schéma général d'un tel changement :

```
dn: champ distinctif
changetype: modify
add: nom d'un champ
nom du champ: valeur du champ
nom du champ: valeur du champ
-
...
replace: nom d'un champ
nom du champ: valeur du champ
nom du champ: valeur du champ
-
...
delete: nom d'un champ
nom du champ: valeur du champ
nom du champ: valeur du champ
-
...
```

Quelques précisions sur ce dernier type de changement :

- ◆ *add* : ajoute un champ avec la valeur donnée pour ce champ si l'enregistrement existe. Si le champ existe déjà, mais avec une autre valeur, une ligne est créée avec le nom du champ et la nouvelle valeur. L'ancienne valeur est conservée.
- ◆ *replace* : La ou les valeurs données vont remplacer toutes les valeurs existantes du champ dont le nom a été indiqué. Si le champ n'existe pas, il est créé avec les valeurs données. Si aucune valeur n'est indiquée, le champ est supprimé.
- ◆ *delete* : Le champ indiqué est supprimé. Si ce champ possède plusieurs valeurs, toutes les valeurs sont supprimées. Pour ne supprimer que certaines valeurs, il faut les indiquer sur la ligne après *delete: nom_de_champ*, et le ou les lignes doivent être de la forme :

```
nom du champ: valeur de l'attribut
```

Exemples :

Voici quelques exemples pour bien comprendre le fonctionnement de la commande `ldapmodify` :

- * Cet exemple permet d'ajouter un nouvel enregistrement (comme la commande `ldapadd`) :

```
grincheux% ldapmodify -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h
grincheux
dn: cn=Titi, o=crihan, c=fr
changetype: add
cn: Titi
telephonenumber: 02 35 11 22 33
objectClass: organizationalPerson
adding new entry cn=Titi, o=crihan, c=fr
ldap_add: Already exists

grincheux%
```

L'enregistrement défini par `cn=Titi, o=crihan, c=fr` est ajouté à l'annuaire. Mais, il existe déjà, c'est pourquoi le message `ldap_add: Already exists` s'affiche.

- * L'exemple ci-dessous permet de changer un attribut d'un enregistrement existant :

```
grincheux.crihan.fr% ldapmodify -D "cn=Delphine Fontaine, o=insa, c=fr" -w s
-p 389 -h grincheux
dn: cn=PEGARD Francis, o=insa, c=fr
changetype: modify
replace: ou
ou: Atelier de Mecanique
^D
modifying entry cn=PEGARD Francis, o=insa, c=fr

grincheux.crihan.fr%
```

Le champ `ou` de l'enregistrement `cn=PEGARD Francis, o=insa, c=fr` change : sa nouvelle valeur est `Atelier de Mécanique`.

- * L'exemple suivant permet d'effectuer deux opérations sur le même enregistrement :

```
grincheux% ldapmodify -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h
grincheux
dn: cn=Titi, o=crihan, c=fr
changetype: modify
delete: telephonenumber
-
add: l
l: Mont Saint Aignan
^D
modifying entry cn=Titi, o=crihan, c=fr

grincheux%
```

Le ou les champs de nom `telephonenumber` sont supprimés et un champ `l` (lieu) est ajouté, sa valeur est `Mont Saint Aignan`.

- * L'exemple suivant permet de modifier un enregistrement en utilisant un fichier grâce à l'option -f.

Le fichier change.ldif contient :

```
dn: cn=Fontaine Delphine, o=crihan, c=fr
changetype: modify
add: ou
ou: ventes
-
delete: l
-
replace: businesscategory
businesscategory: stagiaire
```

La commande sera donc :

```
grincheux% ldapmodify -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h
grincheux -f change.ldif
modifying entry cn=Fontaine Delphine, o=crihan, c=fr

grincheux%
```

Ainsi, dans l'enregistrement *cn=Fontaine Delphine, o=crihan, c=fr* , le ou les champs *l* (lieu) seront supprimés, le ou les champs *businesscategory* (fonction) seront supprimé pour n'avoir que la valeur *stagiaire*, et un champ *ou* (service) sera ajouter et aura la valeur *ventes*.

- * L'exemple suivant permet de supprimer un numéro de téléphone précis :

```
grincheux% ldapmodify -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h
grincheux
dn: cn=Fontaine Delphine,o=crihan,c=fr
changetype: modify
delete: telephonenumber
telephonenumber: 02 35 11 22 33
^D
modifying entry cn=Fontaine Delphine,o=crihan,c=fr

grincheux%
```

Ainsi, le numéro de téléphone *02 35 11 22 33* est supprimé et les autres conservés.

- * Le dernier exemple montre comment utiliser le type de changement *modrdn* :

```
grincheux% ldapmodify -D "cn=Delphine Fontaine, o=crihan, c=fr" -w secret -h
grincheux
dn: cn=Fontaine Delphine,o=crihan,c=fr
changetype: modrdn
newrdn: cn=autre personne
deleteoldrdn: 0
grincheux%
```

Comme *deleteoldrdn* vaut 0, le nouvel enregistrement aura deux valeurs pour le champ *cn* (*cn=Fontaine Delphine* et *cn=autre personne*) en plus des autres champs de l'ancien RDN.

CHAPITRE IX : LES ACCENTS

I) Découverte du problème :

Dans le Make-common, nous avons décommenté la ligne : `LIBLDAP_CHARSET=--DLLDAP_CHARSET_8859="88591"` pour avoir des caractères ISO 8859-1.

En effet, le format ISO 8859-1 nous permet d'avoir des caractères accentués dans notre base.

Ainsi, avec les commandes que l'on exécute directement sous Unix comme `ldapsearch`, permettent de voir les caractères accentués. C'est en fait le client LDAP installé en même temps que le package `ldap-3.3` qui comprend le format ISO 8859-1, format de la base.

Pour voir les accents lors d'une recherche avec la commande "`ldapsearch`", il faut mettre l'option `-B` afin d'afficher les caractères non ascii :

Résultat d'une recherche sans l'option -B :

```
grincheux.crihan.fr% ldapsearch 'cn=prigent*'
cn=PRIGENT Hervé, o=crihan, c=FR
cn=NOT ASCII
sn=PRIGENT
givenname=NOT ASCII
businesscategory=Responsable Service Informatique
mail=Herve.Prigent@crihan.fr
telephonenumber=+33 (0)2 35 59 61 59
facsimiletelephonenumber=+33 (0)2 35 59 61 40
objectclass=top
objectclass=person
objectclass=organizationalPerson
objectclass=inetOrgPerson
```

Résultat de la même recherche avec l'option -B :

```
grincheux.crihan.fr% ldapsearch -B 'cn=prigent*'
cn=PRIGENT Hervé, o=crihan, c=FR
cn=PRIGENT Hervé
sn=PRIGENT
givenname=Hervé
businesscategory=Responsable Service Informatique
mail=Herve.Prigent@crihan.fr
telephonenumber=+33 (0)2 35 59 61 59
facsimiletelephonenumber=+33 (0)2 35 59 61 40
objectclass=top
objectclass=person
objectclass=organizationalPerson
objectclass=inetOrgPerson
grincheux.crihan.fr%
```

Cependant, si on utilise le client Netscape, les accents sont transformés en "?". Netscape ne peut donc pas lire le format ISO 8859-1.

II) Demande d'aide à la liste :

Le format que connaît Netscape est en fait le format utf-8. Pour le savoir, j'ai écrit à la liste ldap :

```
Subject: accents
Date: Wed, 12 Aug 1998 11:22:17 +0200
From: Delphine Fontaine <dfontain@crihan.fr>
Organization: CRIHAN
To: ldap@umich.edu
```

Hello,

```
I have a problem with accents. I have installed Ldap-3.3 with
gdbm-1.7.3. I changed my Make-common to have :
# Otherwise, uncomment this line and set the following options.
#STR_TRANSLATION=-DSTR_TRANSLATION
#
# remove the defines for LDAP client library T.61 character translation
# you do not need.  If you use LDAP_CHARSET_8859, replace the '1' in
"88591"
# with the number of the particular character set you use.  E.g., use
"88594"
# if you use the ISO 8859-4 chracter set.
LIBLDAP_CHARSETS=-DLLDAP_CHARSET_8859="88591"
#
# uncomment one these lines to enable automatic T.61 translation by
default
#LIBLDAP_DEF_CHARSET=-DLLDAP_DEFAULT_CHARSET=LDAP_CHARSET_8859
```

My client LDAP supports accents but Netscape write a "?" instead of an accent. I think that it is not my LDAP installation which is deficient but I don't know how to see accents with Netscape.

Is there anyone having any idea ?

Thanks.

Delphine.



J'ai eu une réponse qui m'a permis d'avancer :

```
Subject: Re: accents
Date: Wed, 12 Aug 1998 15:23:05 +0200
From: Michael Stroeder <michael.stroeder@propack-data.de>
Organization: Propack Data GmbH
To: ldap@umich.edu
References: 1
```

Delphine Fontaine wrote:

```
>
> Hello,
>
> I have a problem with accents.
> [...]
> My client LDAP supports accents but Netscape write a "?" instead of an
> accent. I think that it is not my LDAP installation which is deficient
> but I don't know how to see accents with Netscape.
```

IMHO Netscape expects UTF8-encoded characters. See also:

<http://www.devoto.com/intlchar.html>

Ciao, Michael.

III) Récupération des fichiers nécessaires :

J'ai donc récupéré les fichiers `utf2iso.c` et `iso2utf.c` sur le site <http://www.devoto.com/intlchar.html>. Ces deux fichiers sont écrits en langage C, il faut donc les compiler. (voir en annexe la page `intlchar.html` intitulée International characters).

J'exécute les commandes suivantes pour créer deux exécutables : `utf2iso` et `iso2utf` :

```
grincheux.crihan.fr% gcc utf2iso.c -o utf2iso
grincheux.crihan.fr% gcc iso2utf.c -o iso2utf
```

Nous avons donc deux exécutables :

- `utf2iso` qui permet de faire la conversion du format utf en format iso.
- `iso2utf` qui permet de convertir du texte au format iso en format utf.

Notre base doit donc être convertie au format utf-8 pour que Netscape puisse interpréter les accents.

Pour transformer notre fichier `ldif` qui se nomme `base_ldif` (contenant toutes les données de l'annuaire) au format utf-8, il suffit de taper la commande suivante avant de créer la base avec la commande `ldif2ldb` :

```
grincheux.crihan.fr% cat base_ldif | iso2utf > base_ldif_utf
grincheux.crihan.fr%
```

Ainsi, le fichier `base_ldif_utf` contient la même chose que `base_ldif` mais au format UTF, pas au format ISO. Nous pouvons ensuite créer la base et redémarrer le serveur :

```
grincheux.crihan.fr% ldif2ldb -i base_ldif_utf -f fic.conf
grincheux.crihan.fr% slapd -f fic.conf
```

IV) Le format UTF-8 :

Pour mieux comprendre l'action de ces programmes sur les caractères accentués, voyons un extrait du fichier de type ldif (qui servira d'entrée pour la base) au format ISO 8859-1 et au format UTF-8.

Voici un exemple d'enregistrement contenu dans le fichier base_ldif :

```
dn: cn=CHARTON Beatrice, o=crihan, c=FR
cn: CHARTON Beatrice
sn: CHARTON
givenname: Béatrice
mail: Beatrice.Charton@crihan.fr
businesscategory: Ingénieur Système
telephoneNumber: +33 (0)2 35 59 61 59
facsimileTelephoneNumber: +33 (0)2 35 59 61 40
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
```

Voici le même enregistrement dans le fichier base_ldif_utf, donc au format utf-8 :

```
dn: cn=CHARTON Beatrice, o=crihan, c=FR
cn: CHARTON Beatrice
sn: CHARTON
givenname: B  atrice
mail: Beatrice.Charton@crihan.fr
businesscategory: Ing  nieur Syst  me
telephoneNumber: +33 (0)2 35 59 61 59
facsimileTelephoneNumber: +33 (0)2 35 59 61 40
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
```

Ce format n'est pas tr  s lisible mais    travers le client Netscape, les accents apparaissent    la place des points d'interrogation.

V) Nouveau probl  me :

La recherche sur Netscape permettra de voir les caract  res accentu  s    la place des "?". Cependant, quand on fait une recherche avec le client LDAP int  gr   au package ldap-3.3, les donn  es apparaissent au format utf-8 qui est incompr  hensible.

Pour voir un r  sultat lisible avec la commande "ldapsearch", il faut convertir le r  sultat de la commande au format ISO, ceci gr  ce    l'ex  cutable utf2iso qui se trouve, comme l'ex  cutable iso2utf, dans le r  pertoire /usr/local/home/dfontain/pag. De plus l'option -B de la commande "ldapsearch" est indispensable.

Voici un exemple de recherche pour bien comprendre :

◆ Recherche avec l'option -B sans conversion :

```
grincheux.crihan.fr% ldapsearch -B 'cn=p*'  
cn=PRIGENT Herve, o=crihan, c=FR  
cn=PRIGENT Herve  
sn=PRIGENT  
givenname=HervÃ©  
businesscategory=Responsable Service Informatique  
mail=Herve.Prigent@crihan.fr  
telephonenumber=+33 (0)2 35 59 61 59  
facsimiletelephonenumber=+33 (0)2 35 59 61 40  
objectclass=top  
objectclass=person  
objectclass=organizationalPerson  
objectclass=inetOrgPerson  
grincheux.crihan.fr%
```

◆ Recherche sans l'option -B avec conversion en ISO :

```
grincheux.crihan.fr% ldapsearch 'cn=p*' | utf2iso  
cn=PRIGENT Herve, o=crihan, c=FR  
cn=PRIGENT Herve  
sn=PRIGENT  
givenname=NOT ASCII  
businesscategory=Responsable Service Informatique  
mail=Herve.Prigent@crihan.fr  
telephonenumber=+33 (0)2 35 59 61 59  
facsimiletelephonenumber=+33 (0)2 35 59 61 40  
objectclass=top  
objectclass=person  
objectclass=organizationalPerson  
objectclass=inetOrgPerson  
grincheux.crihan.fr%
```

◆ Recherche avec l'option -B avec la conversion en ISO :

```
grincheux.crihan.fr% ldapsearch -B 'cn=p*' | utf2iso  
cn=PRIGENT Herve, o=crihan, c=FR  
cn=PRIGENT Herve  
sn=PRIGENT  
givenname=Hervé  
businesscategory=Responsable Service Informatique  
mail=Herve.Prigent@crihan.fr  
telephonenumber=+33 (0)2 35 59 61 59  
facsimiletelephonenumber=+33 (0)2 35 59 61 40  
objectclass=top  
objectclass=person  
objectclass=organizationalPerson  
objectclass=inetOrgPerson  
grincheux.crihan.fr%
```

Nous avons le même problème avec les autres clients LDAP existants. En effet, tous les clients ne reconnaissent pas le format UTF-8. Il a donc fallu choisir.

VI) Choix d'un compromis :

On devait choisir entre utiliser le format UTF-8 ou rester en ISO 8859-1. On a finalement opter pour la base en UTF-8 pour pouvoir utiliser Netscape.

Le problème pour le client LDAP fourni avec le package n'était pas important. En effet, des pages Web devant être créées pour faciliter la manipulation de l'annuaire pour les utilisateurs, la conversion UTF-ISO pouvait se faire dans les scripts.

C'est donc ce qui a été fait : les recherches peuvent se faire à travers Internet, grâce à des formulaires, et les noms et prénoms des personnes s'affichent à l'écran avec les caractères accentués, sans que l'utilisateur ne s'aperçoive de la conversion.

Il restait pourtant un problème : certains navigateurs ne permettent pas de mettre des caractères accentués dans les différents champs des formulaires. Il fallait donc trouver un moyen pour que l'utilisateur demandant "André" et celui demandant "Andre" trouvent tous les deux tous les André de l'annuaire.

Nous avons donc choisi de ne pas accentuer le champs servant aux recherches des noms, c'est-à-dire le champs "cn" (common name). Les requêtes à travers le serveur Web sont désaccentuées afin d'effectuer la recherche sans les accents.

CHAPITRE X :

LE SERVEUR WEB

Pour que les recherches et l'utilisation des différentes commandes pour gérer l'annuaire soient plus pratiques, j'ai fait des pages html et des scripts cgi. Ainsi, un utilisateur peut gérer l'annuaire à travers Internet. Cependant, tous les utilisateurs ne pourront pas modifier les données de l'annuaire puisqu'il faudra un mot de passe. Seule la recherche pourra se faire par n'importe quel utilisateur ayant accès à ce site.

I) Installation du serveur :

Un serveur Web interne a été installé sur la machine sur laquelle je travaillais (grincheux.crihan.fr). C'est un serveur Apache qui a été installé.

Ce serveur m'a permis de tester mes pages html et mes scripts cgi avant de les mettre sur le site du CRIHAN et sur l'Intranet de l'INSA de Rouen.

Je ne vais pas décrire ici l'installation du serveur en détail car ce n'était pas l'objet de mon stage. Mais, je vais donner les grandes lignes :

Il faut d'abord récupérer le package sur le site <http://www.apache.org>. Ensuite, il faut exécuter les commandes suivantes :

- configure
- make
- make install

Le serveur est installé mais pour pouvoir l'utiliser, il faut être root, alors Béatrice Charton, mon maître de stage, a changé mes droits d'accès sur l'exécutable httpd qui se trouve dans le répertoire apache/sbin.

II) Emplacement des fichiers :

Les fichiers d'extension « .html » sont situés dans le répertoire apache/share/htdocs et les scripts cgi dans le répertoire apache/share/cgi-bin.

Dans chacun des deux répertoires, j'ai créé un répertoire *modifier* qui contient les fichiers dont il faut contrôler l'accès.

Cette installation était provisoire puisque ce serveur ne servait que pour les tests et n'était accessible qu'en interne.

Les scripts cgi ont été écrit en langage PERL, langage proche du shell permettant un traitement plus simple que le langage C sur les chaînes de caractère.

III) La page d'accueil de l'annuaire :

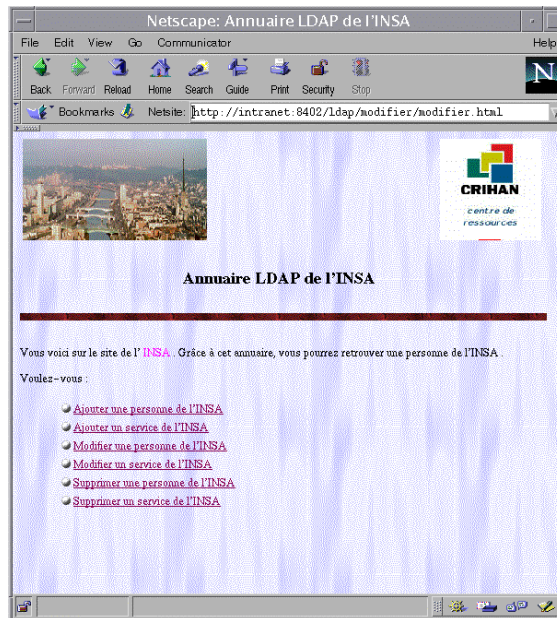
Sur ce serveur, j'ai créé des pages pour pouvoir accéder à l'annuaire du CRIHAN et à celui de l'INSA. En fait, les données étaient rassemblées dans le même fichier LDIF pour les tests.

Mes explications concerneront donc à la fois les fichiers créés pour le CRIHAN et ceux créés pour l'INSA. Nous pourrions remarquer que les fichiers sont très similaires pour les deux.

Voici la page d'accueil de l'annuaire :



La partie administration est réservée aux personnes ayant le droit de modifier les données de l'annuaire pour ajouter, supprimer ou modifier un enregistrement. Elle donne accès à la page suivante :



Parmi les pages créées, certaines sont statiques et certaines sont générées par des scripts cgi. Dans les prochains paragraphes, nous allons voir chacune des opérations que l'on peut effectuer. Le principe de chaque script cgi sera expliqué et des exemples de pages obtenues permettront de mieux comprendre le fonctionnement de ces scripts.

Les explications et les illustrations ne concerneront que les personnes de l'INSA mais c'est le même principe pour les personnes du CRIHAN et les services de l'INSA.

IV) La recherche d'une personne :

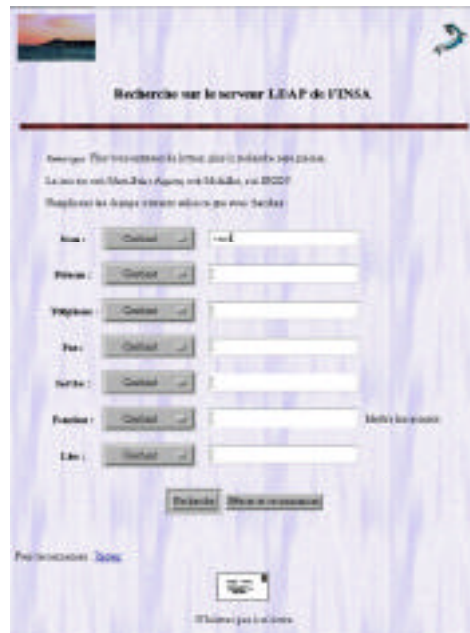
Pour pouvoir rechercher un enregistrement, c'est-à-dire une personne, j'ai fait une page html et un script .

La page html présente un formulaire. Plus il y a de champs remplis, plus la recherche sera précise.

Nous allons prendre un exemple de recherche pour illustrer nos propos : nous allons chercher une personne dont le nom de famille contient la chaîne de caractère "vand".

Voici la page html qui est affichée quand on clique sur le lien "Recherche d'une personne de l'INSA" sur la page d'accueil de l'annuaire :

Remarque: On a rempli le champ Nom par "vand" pour faire notre recherche.



Le script perl, récupère les champs remplis par l'utilisateur et les utilisent pour faire la recherche.

Voici le schéma général de ce que fait ce script "rech_pers_insa.cgi" :

- Il récupère les champs du formulaire de la page html et les valeurs donnés à ces champs par l'utilisateur.
- Désaccentuation de la demande.
- On construit la requête de recherche en fonction des champs remplis.
- Le résultat de la recherche est traité afin d'afficher proprement et avec les noms des champs en français les enregistrement trouvés.

Voici un exemple de résultat sur la recherche faite précédemment :

RÉSULTAT

Votre requête état :

Le Nom : Clément VAND

Voici les personnes trouvées :

Il y a 3 personnes trouvées pour votre requête .

~~~~~

Nom : VANDERPITTE  
Prénom : Charlotte  
Téléphone : 02 35 52 83 21

~~~~~

Nom : VANDERLYNDEN
Prénom : Jenn
Email : Jenn.Vanderlynden@insa-rouen.fr
Téléphone : 02 35 52 83 00
Téléphone : 02 35 52 83 62

~~~~~

Nom : VANDROMME  
Prénom : Dany  
Email : [Dany.Vandromme@insa-rouen.fr](mailto:Dany.Vandromme@insa-rouen.fr)  
Téléphone : 02 32 95 97 40

~~~~~

Pour recommencer : [Retour](#)



C'est le même principe pour les personnes du CRIHAN et pour les services de l'INSA. La principale différence est le nombre de champs disponibles. En effet, les personnes du CRIHAN n'ont pas de champ "service", les services non plus.

V) L'ajout d'une personne de l'INSA :

Pour pouvoir ajouter un enregistrement, c'est-à-dire une personne, j'ai également fait une page html et un script .

La page html présente un formulaire à remplir selon les informations que l'on désire donner à la personne. Les champs nom et prénom doivent être remplis, sinon l'ajout ne peut pas se faire. Tous les autres champs sont optionnels.

Nous allons illustrer nos propos par un exemple : nous allons ajouter une personne dont le nom de famille est "Dupont" et le prénom "Stéphane". Il a un numéro de téléphone qui est "02 35 11 22 33" et une adresse email "Stephane.Dupont@insa-rouen.fr". De plus, on indique que c'est un stagiaire.

Remarque : Cette personne n'existe pas, on s'en sert seulement comme test.

Pour faire cet ajout, on remplit les champs qui se trouvent sur la page html suivante :



The screenshot shows a web browser window with the title 'Ajout d'une personne'. The page content includes a header with the title, a sub-header 'Saisissez l'adresse la plus précise possible', and a list of fields for data entry: 'Nom', 'Prénom', 'Date de naissance', 'Sexe', 'E-mail', 'Adresse', 'Code postal', 'Ville', and 'Autres lieux'. At the bottom, there are two buttons: 'Valider' and 'Retour à l'accueil', and a small icon of a person.

Le script perl, récupère les champs remplis par l'utilisateur et les utilisent pour faire l'ajout de cette personne.

Voici le schéma général de ce que fait ce script "ajout_pers_insa.cgi" :

- Il récupère les champs du formulaire de la page html et les valeurs donnés à ces champs par l'utilisateur.
- **Si** le champ nom ou prénom ou les deux n'est pas remplis, **alors** l'ajout ne se fait pas.

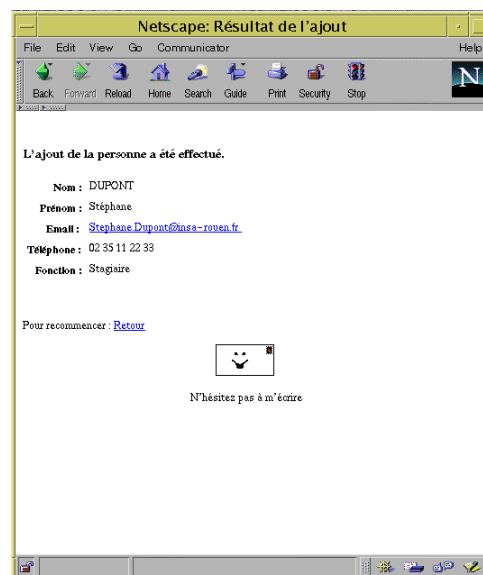
Sinon :

- Il met les champs au bon format (nom en majuscule, prénom en minuscule et la première lettre en majuscule, service en majuscule désaccentué, la fonction et le lieu en minuscule avec la première lettre en majuscules).
- Il cherche la personne dont le nom et le prénom (désaccentués) correspondent à ceux qui ont été donnés (désaccentués).
- **Si** la personne existe, **alors** on affiche sa fiche et un message disant que la personne existe déjà.

Sinon :

- Il construit l'enregistrement à ajouter avec les données entrées par l'administrateur dans le formulaire précédent.
- Il ajoute cet enregistrement.
- Il recherche la personne que l'on vient d'ajouter pour afficher sa fiche. Ceci permet de vérifier que la mise à jour s'est passée correctement.

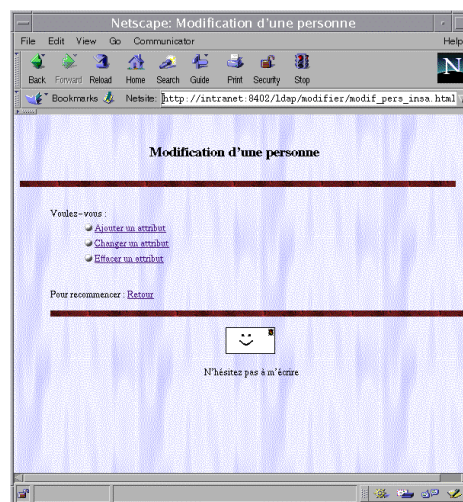
Voici le résultat obtenu en ajoutant Stéphane Dupont, s'il n'existait pas avant :



VI) La modification d'une personne :

Quand on clique sur le lien "Modifier une personne de l'INSA (sur la page réservée à l'administration), on arrive sur une page présentant un nouveau menu. En effet, modifier une personne signifie ajouter, supprimer ou remplacer un champ de la fiche (ou enregistrement) de cette personne.

Voici cette page :

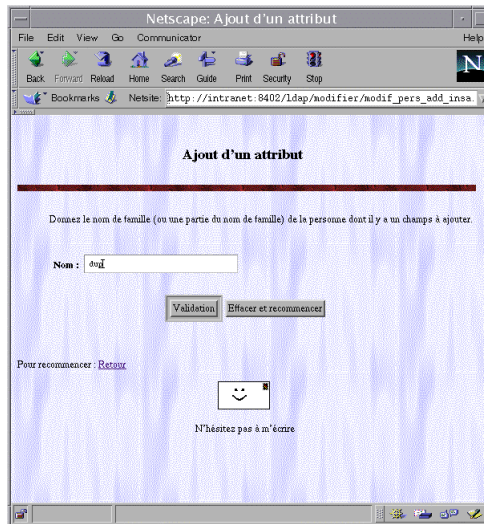


a) Ajouter un champ :

Quand on clique sur le lien "Ajouter un attribut", on arrive sur une page html avec un champ à remplir. Dans ce champ, on indique un nom de famille ou partie du nom de famille de la personne dont on veut ajouter un champ.

Prenons par exemple la personne que l'on a ajouté précédemment. On va lui ajouter un champ "lieu" de valeur "Canteleu".

Voici la page html déjà remplie :

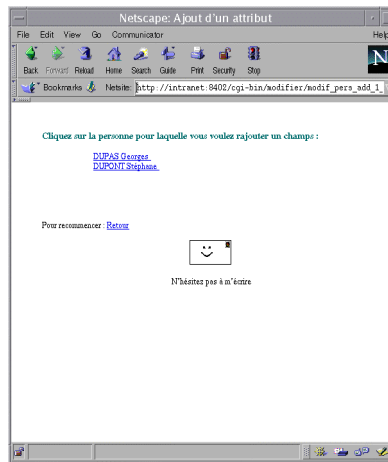


Puis, on valide. Une autre page html est générée par un script perl "modif_pers_add_1_insa.cgi".

Voyons ce que fait ce script :

- Il récupère la chaîne de caractères donnée par l'administrateur (ou utilisateur ayant le droit de modifier le contenu de l'annuaire).
- **Si** le champ n'a pas été rempli, **alors** un message l'indiquant à l'utilisateur s'affiche.
Sinon :
 - Il cherche la ou les personnes dont le nom contient la chaîne de caractères donnée.
 - **Si** au moins une personne correspondante à cette requête existe, **alors** il affiche les noms et prénoms de ces personnes en faisant un lien sur la page suivante, page html générée par un script perl auquel on donne le nom et prénom de la personne (champ "cn") en argument.
Sinon :
 - Un message indiquant que la personne n'existe pas s'affiche.

Ainsi, pour notre exemple, la page suivante s'affiche :

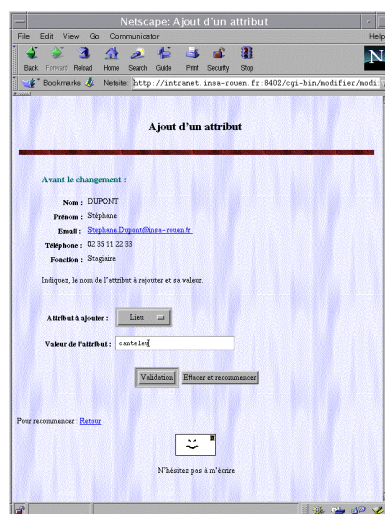


Nous choisissons Dupont Stéphane en cliquant dessus. Un autre script cgi est appelé pour générer un autre page html.

Voyons ce que fait ce script "modif_pers_add_2_insa.cgi" :

- Il récupère le champ "cn".
- Il recherche dans l'annuaire la personne dont le champ "cn" est celui récupéré et il affiche la fiche de cette personne.
- Il crée un formulaire avec un nom de champ que l'on veut ajouter à sélectionner dans une liste contenant les différents champs possibles et la valeur de ce champ à entrer au clavier.

Voici donc cette page dûment remplie :



Quand on valide, on parvient à une dernière page générée par un troisième script "modif_pers_add_3_insa.cgi". Le schéma de ce script est :

- Il récupère le champ "cn" de la personne à laquelle on ajoute un champ.
- **Si** le champ du formulaire n'a pas été rempli, **alors** un message l'indiquant à l'utilisateur s'affiche.

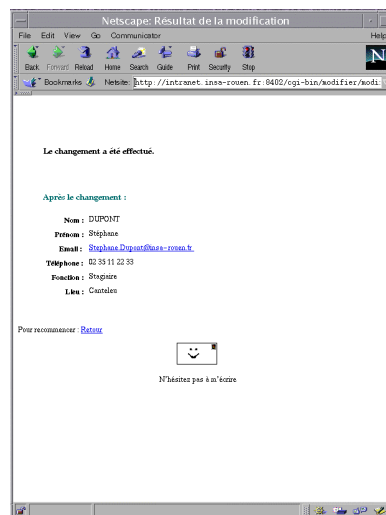
Sinon :

- Il cherche cette personne pour vérifier si elle possède déjà le champ et surtout si pour ce champ elle a déjà la valeur indiquée par l'utilisateur. **Si** c'est le cas, **alors** un message s'affiche pour indiquer que le changement a déjà été fait.

Sinon :

- Il construit la ligne (nom du champ: valeur du champ) à ajouter avec les données entrées par l'administrateur dans le formulaire précédent.
- Il ajoute ce champ.
- Il recherche la personne dont le "cn" avait été transmis et il affiche sa fiche.

Ici, on aura la page suivante :



C'est le même principe pour la suppression ou le changement d'un champ. Voyons les quelques différences :

b) Changement et suppression d'un attribut :

Pour ces deux opérations, on fait la même chose que pour l'ajout d'un champ. En effet, pour supprimer un attribut par exemple, on entre une chaîne de caractère dans le champ nom. Puis, on choisit la personne dont il y a un attribut à enlever dans la liste proposée.

Ensuite, le formulaire avec la liste où on sélectionne le champ à supprimer est créé. La différence est que, dans la liste des attributs que l'on peut supprimer, il n'y a que ceux que possède la personne à modifier. Ceci permet d'avoir moins d'erreurs. C'est la même chose pour le remplacement d'un attribut.

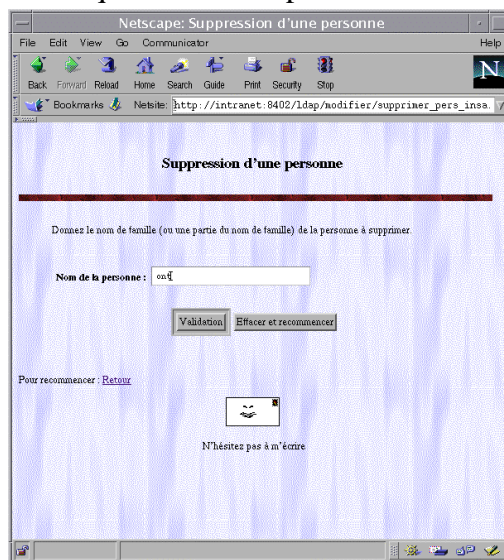
Remarque : On peut supprimer un seul champ pour un enregistrement s'il en a plusieurs du même nom et de valeurs différentes. Il suffit de préciser la valeur de celui qui doit être supprimé. Mais, quand on change un champ, tous ceux du même nom sont supprimés et un nouveau champ de ce nom est créé avec la valeur donnée.

VII) Suppression d'une personne :

Quand on clique sur le lien "Supprimer une personne de l'INSA" sur la page réservée à l'administration, on arrive sur une page html avec un champ à remplir. Dans ce champ, on indique un nom de famille ou partie du nom de famille de la personne dont on veut ajouter un champ.

Pour que les choses soient plus claires, nous allons supprimer "Stéphane Dupont" que l'on avait ajouté précédemment.

Voici donc la page html qu'il fallait remplir :



Puis, on valide. Une autre page html est générée par le script perl "supprimer_pers_1_insa.cgi". Voyons ce que fait ce script :

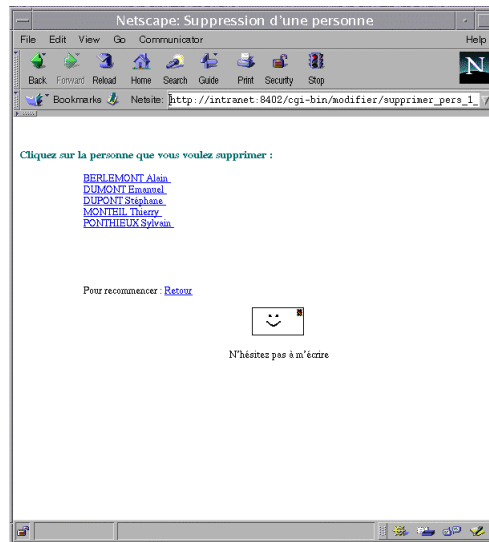
- Il récupère la chaîne de caractères donnée par l'administrateur (ou utilisateur ayant le droit de modifier le contenu de l'annuaire).
- **Si** le champ n'a pas été rempli, **alors** un message l'indiquant à l'utilisateur s'affiche.

Sinon :

- Il cherche la ou les personnes dont le nom contient la chaîne de caractères donnée (préalablement désaccentuée).
- **Si** au moins une personne correspondante à cette requête existe, **alors** il affiche les noms et prénoms de ces personnes en faisant un lien sur la page suivante, page html générée par un script perl auquel on donne le nom et prénom de la personne (champ "cn") en argument.

Sinon :

- Un message indiquant que la personne n'existe pas s'affiche. Ainsi, pour notre exemple, la page suivante s'affiche :

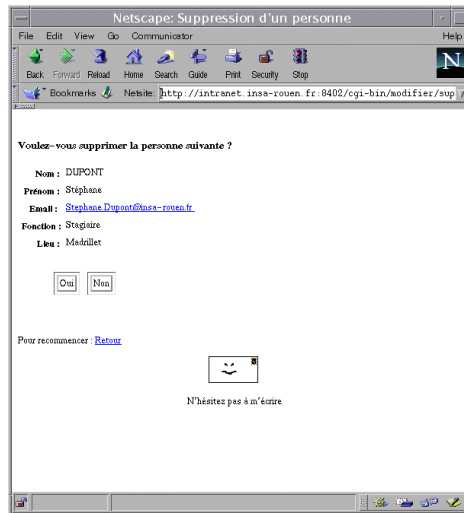


Nous choisissons Dupont Stéphane en cliquant dessus. Un autre script cgi est appelé pour générer un autre page html.

Voyons ce que fait ce script nommé "supprimer_pers_2_insa.cgi" :

- Il récupère le champ "cn".
- Il recherche dans l'annuaire la personne dont le champ "cn" est celui récupéré et il affiche l'enregistrement trouvé.
- Il affiche deux boutons : "oui" et "non" afin de demander une confirmation avant la suppression de la personne affichée.
- **Si** l'utilisateur clique sur "oui", **alors** une page est générée par le script "supprimer_pers_3_insa.cgi".
Sinon, s'il clique sur "non", une page est générée par le script "supprimer_pers_4_insa.cgi".

Voici la page de confirmation obtenue en cliquant sur Stéphane Dupont :



Voyons ce que font les deux scripts "supprimer_pers_3_insa.cgi" et "supprimer_pers_3_insa.cgi" :

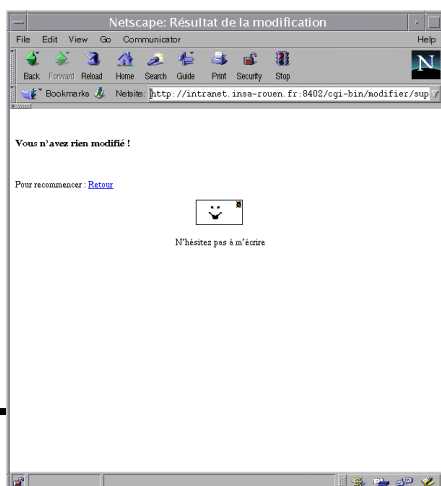
"supprimer_pers_3_insa.cgi":

- Il récupère le champ "cn".
- Il recherche dans l'annuaire la personne dont le champ "cn" est celui récupéré pour vérifier si la personne existe.
- **Si** la personne n'existe pas **alors** un message avertissant l'utilisateur s'affiche.
Si non :
 - On supprime cette personne.

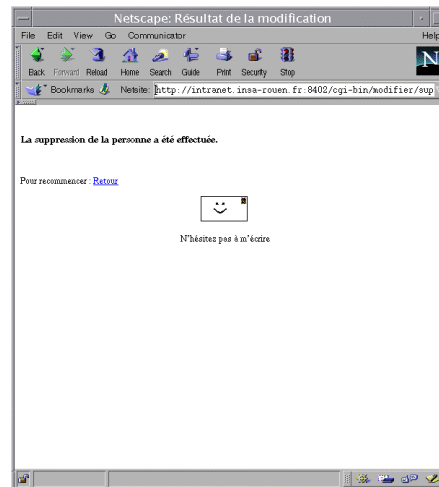
"supprimer_pers_4_insa.cgi":

- Il affiche seulement le message "Vous n'avez rien modifié".

Voici donc les deux pages que l'on peut obtenir :



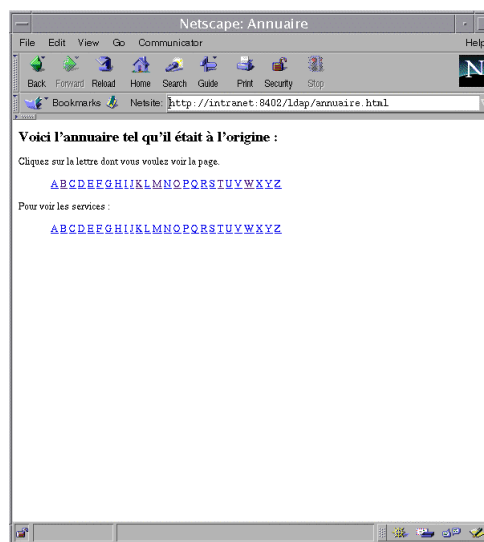
Si on clique sur "Non".
Si on clique sur "Oui".



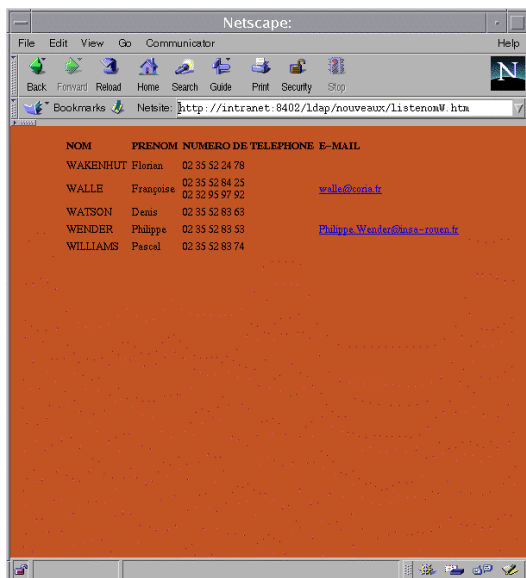
VIII) Le lien "Vers l'ancien annuaire" :

Ce lien visible sur la page d'accueil de l'annuaire de l'INSA permet de voir l'annuaire tel qu'il était avant l'installation et l'utilisation de LDAP. Les fichiers html étaient remplis à la main mais un script permet maintenant de régénérer ces pages html à partir de la base de données (mise à jour facilement par le Web). Ce script sera détaillé dans le chapitre suivant.

Quand on clique sur ce lien, on arrive sur une page html comprenant toutes les lettres de l'alphabet à la fois pour les services et pour les personnes de l'INSA. Les lettres sont en fait des liens vers les pages html constituant annuaire à l'ancien format. Voyons deux exemples :

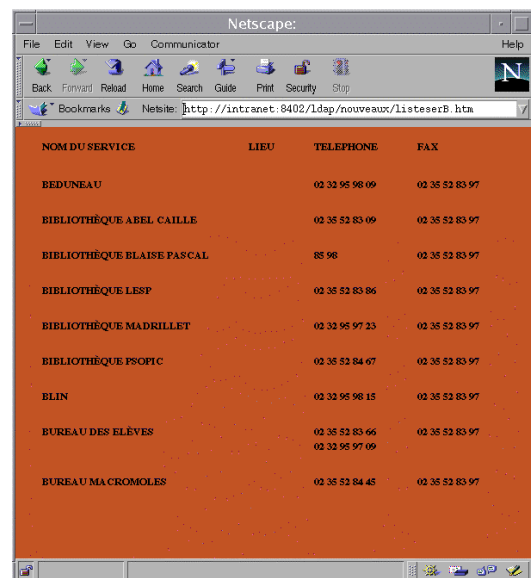


Page d'accueil de l'annuaire à l'ancien format.



NOM	PRENOM	NUMERO DE TELEPHONE	E-MAIL
WAKENHUT	Florian	02 35 52 24 78	
WALLE	Françoise	02 35 52 84 25 02 32 95 97 92	walle@insa.fr
WATSON	Denis	02 35 52 83 63	
WENDER	Philippe	02 35 52 83 53	Philippe.Wender@insa.fr
WILLIAMS	Parcol	02 35 52 83 74	

Page "listenomW.htm" obtenue en cliquant sur la lettre W pour les personnes.



NOM DU SERVICE	LIEU	TELEPHONE	FAX
BEDUNEAU		02 32 95 96 09	02 35 52 83 97
BIBLIOTHEQUE ABEL CAILLE		02 35 52 83 09	02 35 52 83 97
BIBLIOTHEQUE BLAISE PASCAL		85 98	02 35 52 83 97
BIBLIOTHEQUE LESP		02 35 52 83 86	02 35 52 83 97
BIBLIOTHEQUE MADRILLET		02 32 95 97 23	02 35 52 83 97
BIBLIOTHEQUE PSOPTC		02 35 52 84 67	02 35 52 83 97
BLIN		02 32 95 96 15	02 35 52 83 97
BUREAU DES ELEVES		02 35 52 83 66 02 32 95 97 09	02 35 52 83 97
BUREAU MA CROMOLES		02 35 52 84 45	02 35 52 83 97

Page "listeserB.htm" obtenue en cliquant sur la lettre B pour les services.

CHAPITRE XI : **QUELQUES SCRIPTS UTILES**

Voici quelques scripts que j'ai créés pendant le stage. Les trois premiers scripts sont écrits en langage PERL (proche du Shell). Ce langage permet une manipulation de fichiers et de chaînes de caractères plus simple que le langage C. Les deux derniers scripts sont écrits en C-Shell.

Les listing sont en annexes

D) convers.pl :

Ce programme permet de créer un fichier LDIF à partir des pages HTML que m'a donné l'INSA.

Voici un exemple de contenu des fichiers sources des pages HTML.

```
<TR><TD>ANSQUER</TD><TD>Xavier</TD><TD>02 35 52 83 71 ou 8515</TD><TD><A
HREF="MAILTO:ansquer@insa-rouen.fr">*</A></TD></TR>
<TR><TD>ARSON</TD><TD>Michel</TD><TD>02 35 52 84 01</TD><TD><A HREF="MAILTO:
arson@insa-rouen.fr">*</A></TD></TR>
<TR><TD>ARTIGUE</TD><TD>Monique</TD><TD>8534</TD></TR>
<TR><TD>ARZEL</TD><TD>Erwan</TD><TD>02 35 52 24 79</TD></TR>
<TR><TD>Assoc Anciens élèves</TD><TD></TD><TD>02 35 52 83 97 ou 02 32 95 98
05</TD></TR>
```

Pour récupérer les adresses email existantes, j'ai utilisé les fichiers generics et users de l'INSA.

Voici un exemple de contenu du fichiers generics :

```
arson          Michel.Arson<@insa-rouen.fr>
barbas        Valerie.Barbas<@insa-rouen.fr>
barbieux      Janine.Barbieux<@insa-rouen.fr>
briant        Alice.Briant<@insa-rouen.fr>
brulefer      Caroline.Brulefert<@insa-rouen.fr>
cote          Elisabeth.Cote<@insa-rouen.fr>
charron       Axelle.Charron<@insa-rouen.fr>
```

Il va nous permettre de récupérer les adresses emails des personnes présentent dans l'annuaire, si elles en ont une. Mais, devant leur adresse email, c'est leur login et pas leur nom et prénom. Donc, il faut d'abord chercher leur login dans le fichier users.

Exemple de contenu du fichier users :

```
arson:arson@regir
Michel.Arson:arson
barbas:barbas@regir
Valerie.Barbas:barbas
barbieux:barbieux@regir
Janine.Barbieux:barbieux
briant:briant@regir
Alice.Briant:briant
```

Nous pouvons donc obtenir le login d'une personne en ayant son nom et son prénom. En effet, le login se trouve après « prénom.nom ».

Il y a pourtant quelques personnes qui ont une adresse email mais qui ne sont pas dans les fichiers users et generics. Ces personnes sont dans le fichier ficcoria dont voici un exemple de contenu :

```
Garo@insa-rouen.fr           :Annie.Garo@coria.fr
Jean-Noel.Letoulouzan       :letoulouzan@coria.fr
Letoulouzan                  :letoulouzan@coria.fr
Bruno.Maheu@insa-rouen.fr   :maheu@coria.fr
Jean.Maquet@insa-rouen.fr   :maquet@coria.fr
```

Voyons le schéma général du script *convers.pl* :

- Il ouvre le fichier LDIF qui contiendra tous les enregistrements.
- Il crée le premier enregistrement qui définit l'INSA comme organisation.
- **Pour** toutes les lettres de l'alphabet :
 - il ouvre le fichier contenant la liste des personnes et des services (dans les fichiers comme listenomA.htm les services et les personnes sont mélangés) dont le nom commence par la lettre de l'alphabet que l'on traite.
 - **Pour** chaque ligne de ce fichier :
 - Il transforme les é ; etc.. en accents : é, etc...
 - **Si** c'est une ligne du type : <TR><TD>Nom</TD><TD> </TD></TR> contenant divers renseignements, **alors** :
 - **Si** c'est un service (quand il n'y a qu'un nom sans prénom, il considère que c'est un service), **alors** il crée un enregistrement avec tous ces renseignements associés à ce service en les mettant au bon format (champ cn désaccentué, et nom en majuscules) et précise que c'est un service.
 - **Sinon** :
 - C'est une personne (il y a un nom et un prénom), alors, il crée un enregistrement avec tous les renseignements associés à cette personne en les mettant au bon format (champ cn désaccentué, nom en majuscules et prénom en minuscules avec la première lettre en majuscules) et précise que c'est une personne.
 - Remarque : pour l'adresse email, il regarde si elle est indiquée, sinon, il recherche le login dans le fichier users puis recherche l'adresse dans le fichier generics. Si elle n'est toujours pas trouvée, il regarde dans le fichier ficcoria.

Ce programme est long à s'exécuter, alors, il est préférable de le faire une seule fois pour créer la base, puis de mettre à jour l'annuaire grâce aux pages Web et scripts CGI mis en place.

II) lieu.pl :

« lieu.pl » est un script permettant de mettre à jour la base de données de l'INSA en ajoutant les lieux en fonction des numéros de téléphone. Ceci peut se faire pendant que le serveur slapd « tourne ».

Selon les numéros de téléphone, nous pouvons savoir si la personne ou le service ayant ce numéro se trouve sur Mont Saint Aignan, Madrillet ou le site de l'IRCOF :

02 35 52 83 00 à 02 35 52 84 99 : INSA MONT SAINT AIGNAN

02 35 52 24 00 à 02 35 52 24 99 : IRCOF

02 35 52 29 00 à 02 35 52 29 99 : IRCOF

02 35 52 97 00 à 02 35 52 98 49 : INSA MADRILLET

Voyons ce que fait ce script :

- **Pour** toutes les lettres de l'alphabet :
 - il cherche toutes les personnes ayant un numéro de téléphone et dont le nom commençant par la lettre qu'il traite.
 - **Pour** personne trouvée :
 - Il ajoute le champ lieu en fonction du numéro de téléphone.

III) tri.pl :

Ce script permet de régénérer les pages HTML à partir de la base de données. C'est en fait l'inverse de convers.pl.

Schéma de ce script PERL :

- **Pour** toutes les lettres de l'alphabet :
 - il crée un fichier de nom listenomA.htm ou autre suivant la lettre traitée.
 - Il cherche tous les enregistrement dont le champ cn commence par cette lettre.
 - **Si** il n'y a aucun enregistrement trouvé, **alors** il écrit dans le fichier HTML le message « Il n'y a pas de personne ou de service dont le nom commence par la lettre .. »

Sinon :

- **Pour** chaque enregistrement trouvé :
 - Il transforme l'enregistrement en ligne du type :

```
<TR><TD>Nom</TD><TD>Prénom ou rien si c'est un service</TD><TD>téléphone....  
</TD></TR>
```

- Il crée un fichier de nom listeserA.htm ou autre suivant la lettre traitée. Cette page listera tous les services.
- Il cherche tous les services dont le nom commence par cette lettre.

- Si il n'y a aucun enregistrement trouvé, **alors** il écrit dans le fichier HTML le message « Il n'y a pas de service dont le nom commence par la lettre .. »

Sinon :

- **Pour** chaque enregistrement trouvé :
 - Il transforme l'enregistrement en ligne du type :

```
<TR><TD>Nom</TD><TD>Prénom ou rien si c'est un service</TD><TD>téléphone....</TD></TR>
```

- on cherche les personnes appartenant à cette association.
- **Si** il y en a , on transforme l'enregistrement en une ligne HTML.

IV) ldbm2ldif:

Ce script en C-Shell permet d'arrêter le serveur slapd et de générer un fichier LDIF en format ISO que l'on pourra modifier puisqu'il sera compréhensible pour nous. Ce programme s'utilise ainsi :

```
ldbm2ldif <nom du fichier LDIF à créer>
```

Ce script utilise la commande `ldbmcat`. Cette commande est comprise dans le package `ldap`. Elle permet de convertir un fichier `id2entry.gdbm` (qui contient la base de données) en fichier LDIF.

La ligne de commande est :

```
ldbmcat [-n] <filename>
```

L'option `-n` est facultative mais permet de ne pas afficher le numéro des enregistrements.

Quand on veut se servir du fichier LDIF obtenu pour reconstruire la base, il faut faire attention à ne pas avoir deux lignes vides qui se suivent. Le script `ldbm2ldif` permet de ne pas avoir à s'inquiéter de ce problème.

En effet, voici ce que fait ce programme :

- Il vérifie si un nom de fichier bien été entré en argument. Si ce n'est pas le cas, il sort.
- Si le fichier existe déjà, alors il demande une confirmation pour l'écraser ou un autre nom de fichier.
- Il arrête le serveur.
- Il crée le fichier LDIF avec la commande `ldbmcat` et supprime les doubles lignes vides pour n'en avoir qu'une entre chaque enregistrement.
- Il met ce fichier en format ISO.

V) genere ldbm:

Ce dernier script permet de redémarrer le serveur après avoir modifier le fichier LDIF. En effet, grâce au programme précédent, on peut modifier directement le fichier LDIF. Ensuite, il faut recréer la base et redémarrer le serveur. C'est le rôle du script « genere_ldbm ».

La ligne de commande est :

```
genere_ldbm <nom du fichier LDIF contante les enregistrements>
```

Voici le schéma :

- Il vérifie si un nom de fichier est bien donné en argument.
- Il met ce fichier en format UTF-8.
- Il crée la base à partir de ce fichier.
- Il redémarre le serveur.

Pour ces deux derniers scripts, les chemins des exécutable sont à changer selon l'endroit où ils se situent sur la machine.

CONCLUSION

Ce stage m'a permis d'acquérir une certaine méthode de travail et a été très positif pour ma formation.

L'annuaire, les pages HTML et les scripts PERL permettant de le mettre à jour seront effectivement mis en place sur le serveur du CRIHAN dès disponibilité de la machine dédiée à cet usage. Les utilisateurs d'Internet pourront ainsi rechercher une personne, un numéro de téléphone ou une adresse email à travers le Web.

Certes, aujourd'hui l'annuaire du CRIHAN ne contient qu'une douzaine de personnes mais il est envisagé d'y faire figurer les quelques centaines d'utilisateurs ayant accès aux ressources informatiques du centre³.

En ce qui concerne l'INSA, l'annuaire est déjà disponible sur l'INTRANET (<http://intranet:8402>). Les élèves et tout le personnel de l'INSA peuvent le consulter. Il contient tous les éléments qui m'ont été communiqués sur le personnel et pourra être complété par les données sur les élèves.

Enfin, ce travail m'a permis d'apprendre à connaître et utiliser tous les outils de communication de l'Internet, notamment les listes de diffusions qui m'ont beaucoup apporté pour la recherche et l'échange d'informations.

J'ai pu également mettre en pratique et développer certains des langages évoqués en cours, comme HTML, CGI ou PERL.

J'ai surtout appris à connaître et à utiliser LDAP, protocole du domaine public, qui, par ses performances et sa souplesse, est appelé à un fort développement sur les réseaux.

³ Ceci suppose une information précise aux personnes et leur accord exprès, conformément aux dispositions de la loi "Informatique et Libertés" (Réf : <http://www.cnil.fr>)

GRILLE DE DEROULEMENT DU STAGE

Voici, semaine après semaine, le déroulement du stage avec l'avancement du projet.

Semaine 1 Du 20 juillet au 27 juillet 1998	Découverte de LDAP et recherches d'informations sur Internet. Début de l'installation de LDAP et rencontre des premiers problèmes.
Semaine 2 Du 28 juillet au 2 août 1998	Suite et fin de l'installation. Tests des différentes commandes sur une petite base créée à partir d'un fichier LDIF contenant deux ou trois enregistrements. Test du client Netscape sur cette petite base. Récupération des pages HTML de l'ancien annuaire de l'INSA.
Semaine 3 Du 3 août au 9 août 1998	Découverte des problèmes d'accents et recherche de la solutions. Apprentissage du langage HTML. Installation d'un serveur WEB. Récupération des fichiers users et generics de l'INSA.
Semaine 4 Du 10 août au 16 août 1998	Inscription à la liste LDAP. Apprentissage du langage PERL et scripts CGI. Conversion des pages HTML de l'INSA en un seul fichier LDIF avec un script PERL. Utilisation de la commande ldapsearch par un script PERL. Résolution des problèmes d'accents (utilisation du format UTF-8)
Semaine 5 Du 17 août au 23 août 1998	Essais avec deux serveurs slapd sur deux ports différents pour utiliser deux bases (une pour l'INSA, une pour le CRIHAN) mais finalement, utilisation d'un seul fichier LDIF rassemblant les informations des deux. Début d'installation de LDAP sur la machine ns de l'INSA mais abandon (l'installation serait mieux sur la machine intranet). Création du script pour régénérer les pages HTML de l'ancien annuaire à partir de la base de donnée du nouvel annuaire.
Semaine 6 Du 24 août au 30 août 1998	Création de tous les scripts et pages HTML permettant de mettre à jour l'annuaire à partir du serveur WEB pour le CRIHAN et pour l'INSA (personnes et services) et débogage.

Semaine 7 Du 31 août au 6 Septembre 1998	Tests sur les scripts et modifications de ceux-ci. Création des scripts <code>ldbm2ldif</code> et <code>genere_ldbm</code> qui permettent d'arrêter le serveur en générant le fichier LDIF (<code>ldbm2ldif</code>), modifier le fichier LDIF directement et redémarrer le serveur en ayant régénérer la base (<code>genere_ldbm</code>).
Semaine 8 Du 7 au 13 Septembre 1998	Début de la rédaction du rapport de stage. Débuggage des scripts. Mise des fichiers nécessaires à l'installation de LDAP sur la machine Intranet de l'INSA.
Semaine 9 Du 14 au 20 Septembre 1998	Installation de LDAP sur la machine Intranet de l'INSA. Dernier tests sur les scripts créés pendant la semaine 6. Fin de la rédaction du rapport de stage.

ANNEXE :

Vous trouverez ici :

- Un résumé sur l'emplacement des différents fichiers et la syntaxe des commandes importantes.
- les fichiers Make-common et ldapconfig.h.edit qui font partie du package LDAP.
- la page intlchar.html intitulée "International characters".
- les listing des fichiers HTML et scripts PERL vus dans le chapitre « Le serveur WEB ». Il n'y a que ceux pour l'INSA car c'est la même chose pour le CRIHAN :
 - rech_insa.cgi et rech_serv_insa.cgi : programmes permettant de faire des recherches dans l'annuaire.
 - ajout_pers_insa.cgi et ajout_serv_insa.cgi : programmes permettant d'ajouter une personne pour le premier et un service pour le second.
 - modif_pers_add_1_insa.cgi, modif_pers_add_2_insa.cgi, modif_pers_add_3_insa.cgi : permettent d'ajouter un champ à l'enregistrement d'une personne.
 - modif_pers_delete_1_insa.cgi, modif_pers_delete_2_insa.cgi et modif_pers_delete_3_insa.cgi : permettent de supprimer un champ à l'enregistrement d'une personne.
 - modif_pers_replace_1_insa.cgi, modif_pers_replace_2_insa.cgi et modif_pers_replace_3_insa.cgi : permettent de remplacer un champ à l'enregistrement d'une personne.
 - modif_serv_add_1_insa.cgi, modif_serv_add_2_insa.cgi et modif_serv_add_3_insa.cgi: permettent d'ajouter un champ pour un service.
 - modif_serv_delete_1_insa.cgi, modif_serv_delete_2_insa.cgi, modif_serv_delete_3_insa.cgi : permettent de supprimer un champ pour un service.
 - modif_serv_replace_1_insa.cgi, modif_serv_replace_2_insa.cgi, modif_serv_replace_3_insa.cgi : permettent de changer un champ.
 - supprimer_pers_1_insa.cgi, supprimer_pers_2_insa.cgi, supprimer_pers_3_insa.cgi, supprimer_pers_4_insa.cgi : permettent de supprimer une personne de l'annuaire.
 - supprimer_serv_1_insa.cgi, supprimer_serv_2_insa.cgi, supprimer_serv_3_insa.cgi, supprimer_serv_4_insa.cgi : permettent de supprimer un service dans l'annuaire.
 - Les différentes pages HTML.
- les listings des scripts vus dans le chapitre « Quelques scripts utiles » :
 - convers.pl : transforme les pages HTML en fichier LDIF.
 - lieu.pl : met à jour la base en ajoutant les lieux pour l'INSA.
 - tri.pl : régénère les pages HTML.
 - ldbm2ldif : génère le fichier LDIF à partir de la base et arrête le serveur.
 - genere_ldbm : régénère la base et redémarre le serveur.

FICHIERS ET COMMANDES IMPORTANTES

Voici un résumé pour retrouver rapidement les différents fichiers évoqués dans le rapport (ce sont les emplacements des fichiers sur la machine intranet.insa-rouen.fr) :

Nom du répertoire	Contenu du répertoire
<i>/src/intranet/cgi-bin</i> et <i>/src/intranet/cgi-bin/modifier</i>	Tous les programmes CGI écrits en PERL permettant de faire des recherches dans l'annuaire et de le mettre à jour.
<i>/src/intranet/ldap</i> et <i>/src/intranet/ldap/modifier</i>	Toutes les pages html permettant d'accéder aux programmes CGI.
<i>/src/intranet/ldap/originaux</i>	Pages "html" de l'ancien annuaire de l'INSA.
<i>/src/intranet/ldap/nouveaux</i>	Exécutable tri.pl Pages "html" créées à partir de l'annuaire LDAP grâce à cet exécutable.
<i>/usr/local/ldap/SERVER</i>	Fichiers base et base_ldif_utf , contenant les enregistrements, l'un au format ISO et l'autre au format UTF-8. Le fichier de configuration fic.conf dont on se sert à la création de la base et au démarrage du serveur slapd.
<i>/usr/local/ldap/SERVER/bin</i>	Exécutables genere_ldbm , iso2utf , ldbm2ldif , et utf2iso .
<i>/usr/local/ldap/bin</i>	Entre autres, exécutables ldapadd , ldapdelete , ldapmodify , ldapmodrdn , ldapsearch .
<i>/usr/local/ldap/etc</i>	Entre autres, exécutables ldif2ldbm , ldbmcat , et slapd .

Les commandes importantes :

Reconstruire la base de données à partir du fichier LDIF "base_ldif_utf" qui est au format UTF-8 :

```
ldif2ldb -f fic.conf -i base_ldif_utf
```

Redémarrer le serveur slapd :

```
slapd [-d 1] -f fic.conf
```

Générer un fichier LDIF au format ISO à partir de l'annuaire et arrêter le serveur :

```
ldb2ldif nom_du_fichier_ldif_à_créer
```

Regénérer la base à partir d'un fichier ldif au format ISO et redémarrer le serveur :

```
genere_ldb nom_du_fichier_ldif
```