Integrazione Sistemistica con LDAP

Simone Piccardi
piccardi@truelite.it

Questa documentazione libera è stata sviluppata all’interno delle attività formative effettuate da Truelite S.r.l. Il materiale è stato finanziato nel corso della realizzazione dei corsi erogati dall’azienda, e viene messo a disposizione di tutti sotto licenza GNU FDL.

Questo testo, insieme al resto della documentazione libera realizzata da Truelite S.r.l., viene distribuito su internet all’indirizzo:

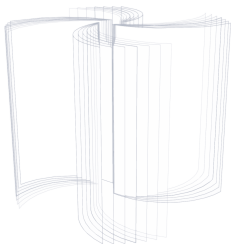
http://svn.truelite.it/truedoc

dove saranno pubblicate nuove versioni ed aggiornamenti.

Società italiana specializzata nella fornitura di servizi, consulenza e formazione esclusivamente su GNU/Linux e software libero.

Per informazioni:

Truelite S.r.l
Via Monferrato 6,
50142 Firenze.
Tel: 055-7879597
Fax: 055-7333336
e-mail: info@truelite.it
web: http://www.truelite.it



Indice

1 Integrazione sistemistica con LDAP **1**

1.1 Il protocollo LDAP. 1
1.1.1 Una visione di insieme 1
1.1.2 La strutturazione dei dati di LDAP 2
1.1.3 I file in formato LDIF 5
1.1.4 Il meccanismo delle interrogazioni su LDAP 6
1.2 Installazione e configurazione di base di *OpenLDAP* 8
1.2.1 Installazione ed avvio del servizio 8
1.2.2 La configurazione di base del client 13
1.2.3 La configurazione di base del server 15
1.2.4 Creazione e manutenzione di un elenco 21
1.2.5 Interrogazioni e ricerche su LDAP 24
1.3 Configurazioni avanzate 27
1.3.1 Autenticazione e controllo degli accessi 27
1.3.2 La configurazione per l'uso di SSL/TLS 33
1.3.3 Gli *overlay* 39
1.3.4 Ottimizzare le prestazioni 40
1.4 La replicazione 40
1.4.1 Il meccanismo della replicazione 41
1.4.2 La replicazione con `slurpd` 42
1.4.3 La replicazione con *syncrepl* 45
1.4.4 Utilizzo avanzato di *syncrepl* 49
1.5 La centralizzazione della gestione di utenti e gruppi 50
1.5.1 La creazione della infrastruttura dei dati 50
1.5.2 La configurazione del *Name Service Switch* su LDAP 56
1.5.3 La configurazione di PAM per l'autenticazione su LDAP 59
1.6 L'integrazione dei servizi su LDAP 62
1.6.1 La gestione dell'indirizzario 63
1.6.2 Apache e LDAP 64
1.6.3 Postfix e LDAP 67
1.6.4 Samba e LDAP 68
1.6.5 Squid e LDAP 76

A GNU Free Documentation License **81**

A.1 Applicability and Definitions 81
A.2 Verbatim Copying 82
A.3 Copying in Quantity 82
A.4 Modifications 83
A.5 Combining Documents 84
A.6 Collections of Documents 85

A.7 Aggregation With Independent Works 85
A.8 Translation 85
A.9 Termination 85
A.10 Future Revisions of This License 85

Capitolo 1

Integrazione sistemistica con LDAP

1.1 Il protocollo LDAP.

In questa prima sezione faremo una introduzione generale su LDAP, trattando le caratteristiche essenziali del protocollo, la strutturazione dei dati, le varie convenzioni adottate ed il formato di interscambio in cui questi vengono usualmente tradotti.

1.1.1 Una visione di insieme

Il nome LDAP è l'acronimo di *Lightweight Directory Access Protocol*; si tratta di un protocollo che permette di accedere ad un servizio di *elenco*[1] in versione elettronica, analogo a quello che potrebbe essere un elenco telefonico, ma generalizzato nei contenuti. In sostanza si tratta di un database, ma il cui scopo principale è la ricerca di informazioni, e non la gestione (nel senso di registrazione ed aggiornamento continuo) delle stesse.

Per capire l'utilità di un servizio generico di questo tipo consideriamo come in un sistema GNU/Linux siano già presenti molteplici elenchi di informazioni: la lista degli utenti, quella dei gruppi, quella delle porte TCP e UDP, ecc.[2] In generale si tende a distinguere fra quelli che sono i servizi di elenco *locali*, come la lista degli utenti di `/etc/passwd`, e quelli che invece sono *globali*, come la lista delle corrispondente fra nomi a dominio e numeri IP fornite dal DNS.

La presenza di un servizio che permetta di mantenere diversi tipi di informazione e di recuperarli in maniera efficiente sulla base di criteri di ricerca generici viene allora a costituire uno strumento fondamentale in tutti quei casi in cui si debbano integrare fra loro sistemi diversi che necessitano di accedere a informazioni comuni, che a questo punto possono essere mantenute in maniera centralizzata all'interno di questo servizio.

I servizi di *elenco* sono stati standardizzati (partendo dall'idea dell'elenco telefonico) a livello internazionale nello standard ISO[3] X.500, che prevede sia un modello di dati che un protocollo di accesso (il DAP, acronimo di *Directory Access Protocol*); LDAP parte dal modello di dati di X.500 ma implementa un protocollo di accesso semplificato basato esclusivamente su TCP/IP, che è stato standardizzato dall'RFC 2251. Oltre a questo sono state semplificate alcune rappresentazioni dei dati e cancellate alcune funzionalità oscure e poco utilizzate.

Il protocollo LDAP è strutturato sul classico modello client-server, in cui in server mantiene le informazioni dell'elenco ed i client eseguono le interrogazioni e le ricerche. Inoltre in maniera analoga a quanto accade per i DNS qualora un server non possieda le informazioni richieste, può trasferire le richieste ad un altro server. L'organizzazione ad albero (che approfondiremo in

[1] non ho trovato una traduzione migliore del termine *directory* in questo contesto.

[2] buona parte di questi sono gestiti dal sistema del *Name Service Switch*, vedremo come potranno essere portati su LDAP in sez. 1.5.2.

[3] la *International Organization for Standardization*, una organizzazione non governativa dedicata allo sviluppo di standard, costituita da una rete di istituti nazionali, che invece sono spesso di origine governativa.

sez. 1.1.2) permette allora di integrare le informazioni locali all'interno di una struttura globale di tutti i servizi di elenco.

Il protocollo è costituito da una parte che definisce l'organizzazione dei dati (chiamata *Data model*) che permette di stabilire come viene rappresentata l'informazione, uno schema di assegnazione dei nomi (detto *Naming model*) che identifica il singolo dato mantenuto nel sistema, ed infine una modalità di accesso ai dati con tanto di meccanismi molto dettagliati, come vedremo in sez. 1.3.1, per il controllo degli accessi.

Il protocollo invece non dice niente rispetto alle modalità specifiche in cui i dati vengono memorizzati, tratta solo la loro strutturazione astratta, e le modalità con cui possono essere compiute le operazioni previste dal protocollo (ricerca, lettura o scrittura) sugli stessi; l'implementazione è lasciata al funzionamento del singolo server.

Per la natura stessa del servizio le informazioni mantenute in un server LDAP sono per lo più lette, vengono scritte o aggiornate soltanto in maniera occasionale; per questo motivo non sono necessari i complessi meccanismi di *roll-back*[4] e sincronizzazione usualmente presenti nei più comuni database relazionali. Inoltre l'organizzazione delle informazioni è di tipo descrittivo e non relazionale e la struttura è ad albero e non a tabelle; nonostante queste differenze fondamentali con quella che è l'accezione più comune del termine,[5] faremo comunque riferimento alle informazioni mantenute in un server LDAP chiamandole *database*.

Un database LDAP pertanto deve essere ottimizzato per fornire risposte rapide in lettura e nelle ricerche, per essere facilmente replicato, e per supportare la distribuzione di carico su più server. Per questo motivo il protocollo, contrariamente a quanto è richiesto per i database relazionali, considera come accettabili delle inconsistenze temporanee nelle informazioni.

1.1.2 La strutturazione dei dati di LDAP

La caratteristica fondamentale del modello di dati usato da LDAP è la sua struttura ad albero (si parla infatti di DIT o *Data Information Tree*); le informazioni sono infatti mantenute in una gerarchia di *oggetti* o *voci* (in inglese sono chiamati *entries*) a partire da un oggetto o voce iniziale che costituisce la radice dell'albero (identificato con i termini *base* o *suffix*), con i successivi oggetti "*contenuti*"[6] all'interno di altri oggetti del livello precedente. Il protocollo permette in questa maniera di suddividere le informazioni in maniera gerarchica, ed eventualmente di tenerle separate su diversi alberi, che in generale possono stare su server diversi, gestendo in maniera naturale la distribuzione dei dati.

Si parla di oggetti e classi perché la strutturazione dei dati di LDAP segue il paradigma della programmazione ad oggetti: ciascun oggetto è una *istanza* di una o più *classi*, e può contenere i valori di uno o più degli *attributi* che sono definiti all'interno delle classi da cui deriva. La strutturazione dell'informazione è infatti basata sulla cosiddetta *objectClass*, una sorta di "*meta-classe*" che viene usata per definire le singole classi assegnando loro un nome che le identifichi e definendo gli attributi che le compongono. In questo modo il modello dei dati consente anche di utilizzare anche il concetto di ereditarietà tipico della programmazione ad oggetti, e costruire delle classi derivate a partire da una classe di base, estendendone gli attributi.

Si rammenti comunque che sono le singole voci nell'albero che contengono i dati reali, specificati dai valori dei rispettivi attributi: una *objectClass* definisce solo quali sono gli attributi possibili, quali di questi sono necessari, quali sono i loro nomi ed il tipo di dato che contengono

[4] si chiama così quella funzionalità che consente di finalizzare una serie di operazioni di modifica dei contenuti in un'unica volta, garantendo la possibilità, fintanto che non la si esegue, di ritornare allo stato precedente.

[5] quella che porta a identificare i database con i database relazionali interrogabili con il linguaggio SQL, ignorando che questo è solo uno dei tanti modi in cui è possibile organizzare una base di dati.

[6] in realtà i singoli oggetti di per sé non contengono altri oggetti, ma solo gli *attributi* della *classe* cui appartengono, parlare di *contenuto* è solo un modo comodo di rappresentare la struttura ad albero per cui sotto un oggetto possono situarsene degli altri che a loro volta possono averne altri al di sotto, ecc.

(stringa, valore numerico, ecc.). Inoltre ogni oggetto presente nell'albero possiede una serie di attributi "`objectClass`" che vengono definiti automaticamente al nome di ciascuna delle classi da cui esso deriva.[7]

Come ad ogni classe, anche ad ogni attributo viene associato un nome mnemonico che lo identifica; inoltre nella sua definizione viene anche specificato da quale tipo di dati questo può essere costituito: due esempi sono il *Common Name*, identificato dalla stringa "`cn`" che specifica un nome, come `Simone Piccardi`, o l'indirizzo di posta elettronica, identificato dalla stringa "`mail`" che contiene un indirizzo del tipo `piccardi@truelite.it`.

Buona parte della standardizzazione di LDAP consiste appunto nella definizione di una serie di *objectClass* e dei relativi attributi; ad esempio una delle classi di base è la cosiddetta *Organizational Unit* (la classe che definisce una unità organizzativa, definita dalla *objectClass* `organizationalUnit`). Questa prevede l'attributo obbligatorio "`ou`", che contiene il nome dell'unità. In questo modo usando la struttura gerarchica si potranno suddividere opportunamente le informazioni all'interno dell'albero inserendo altre voci al di sotto di una voce di questa classe, come altre *Organizational Unit* o gli oggetti di altre classi dotate degli attributi adatti a contenere i dati delle singole persone.

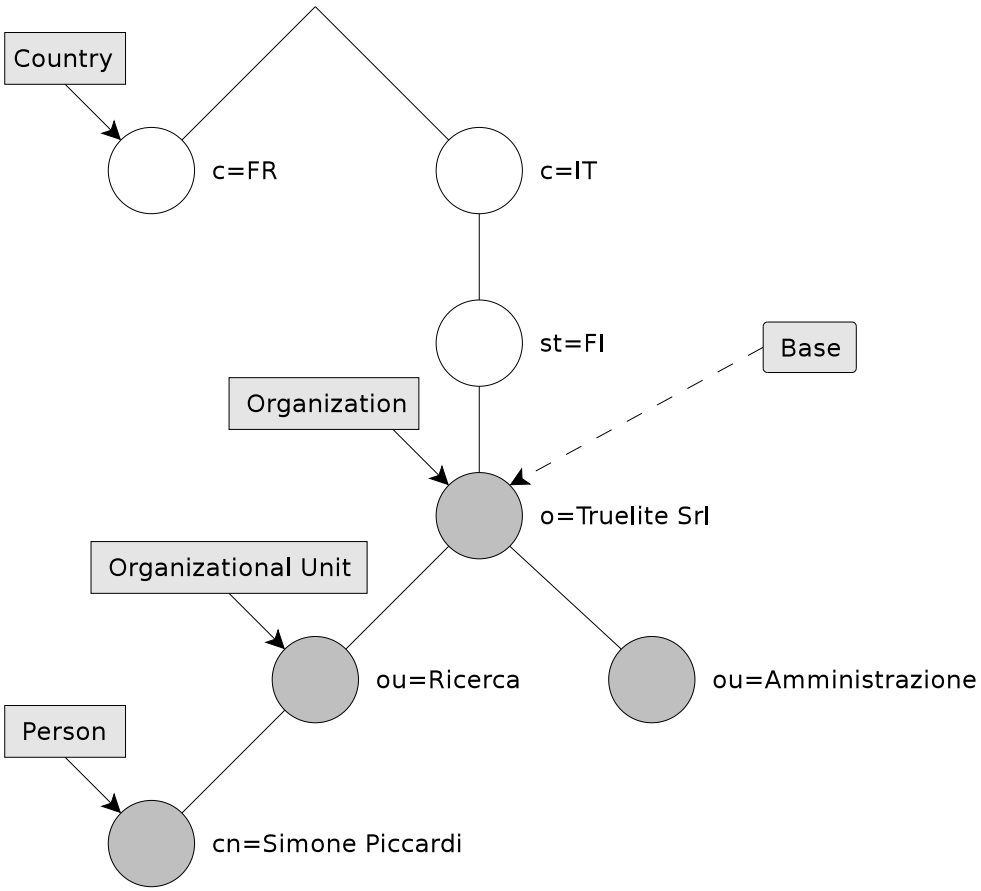


Figura 1.1: Strutturazione di un albero LDAP su base geografica.

La struttura ad albero dei dati di un server, oltre a consentire una suddivisione gerarchica delle informazioni in esso contenute, permette anche di inserire i vari alberi dei singoli server all'interno di una organizzazione dei dati *globale* (in maniera analoga alla struttura dei nomi a dominio), in cui si possono riunire in maniera coerente alberi diversi sotto una radice unica.

[7] l'attributo viene cioè generato automaticamente dal sistema tutte le volte che si fa derivare un oggetto da una determinata classe, ed ovviamente il suo nome è riservato.

Questo ovviamente richiede che la radice usata dai singoli server sia scelta con un criterio coerente che consenta questa integrazione; di questi criteri ne esistono in sostanza due.

Il primo criterio deriva direttamente dallo scopo originale di LDAP, che era quello di fornire un database distribuito, analogo a quello del DNS, che costituisse l'analogo dell'elenco del telefono su internet, contenente informazioni riguardo nomi, indirizzi e dati specifici di società e persone. In questo caso le informazioni sono gerarchizzate su base geografica: alla radice dell'albero stanno allora degli oggetti *Country* che specificano il paese con l'attributo "`c`", sotto i quali poi si situeranno ulteriori oggetti di specificazione geografica più limitata, come "`st`" che indica lo *State*,[8] o "`l`" che indica la località,[9] ed infine "`o`" che indica una organizzazione.[10] Un esempio di questo criterio di strutturazione è riportato in fig. 1.1, ma oggi esso è praticamente in disuso.

Il secondo criterio è invece ripreso direttamente dall'albero dei domini del DNS, di cui LDAP ricalca la filosofia anche per quanto riguarda la distribuzione dell'informazione. In questo caso alla radice dell'albero stanno sempre degli oggetti *Domain Component*, (la cui *objectClass* è `dcObject`), identificati tramite il valore del rispettivo attributo obbligatorio "`dc`", ed inseriti uno sotto l'altro nell'albero in modo da replicare la struttura dei domini su internet. Le successive classificazioni ripartono dal livello di organizzazione o unità organizzativa già visto nella precedente struttura. Un esempio di questa strutturazione è riportato in fig. 1.2.

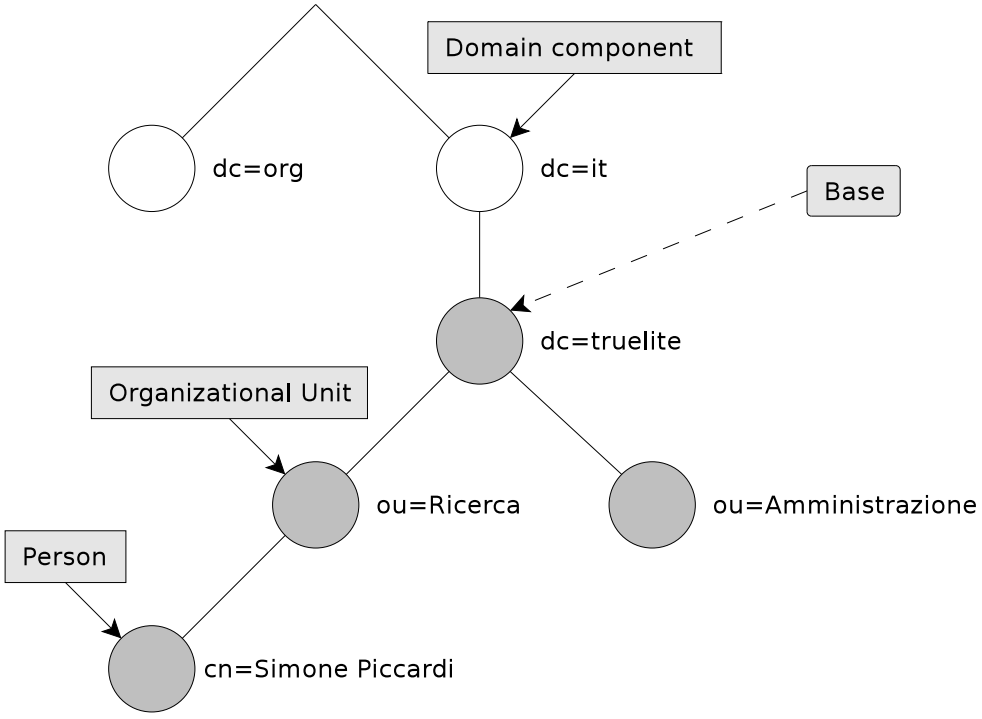


Figura 1.2: Strutturazione di un albero LDAP sulla base dell'albero dei domini.

All'interno dell'albero dei dati (indipendentemente da quale delle due precedenti strutturazioni si è scelta) ogni oggetto è identificato univocamente dal cosiddetto *Distinguished Name* (che in seguito abbrevieremo spesso in DN);[11] questo altro non è che il percorso da fare a partire dalla

[8] essendo nato in ambito statunitense la suddivisione geografica successiva è stata da "*nazione*" (gli USA) a singolo "*stato*" (il Texas); in ambito europeo in genere questo campo indica la provincia o viene omesso.

[9] città, o più in generale il comune.

[10] si sono identificati gli oggetti col nome dell'attributo obbligatorio che essi prevedono, piuttosto che con quello della rispettiva *objectClass*, dato che sono i valori di questi nomi a referenziare i singoli oggetti nell'albero.

[11] il formato dei *Distinguished Name* è definito dall'RFC 2253, che si può consultare per avere tutti i dettagli al proposito.

radice dell'albero, in cui si specificano, sulla base del loro *nome relativo* (detto anche *Relative Distinguished Name*, o RDN) i singoli oggetti da attraversare per arrivare a quello voluto.

Il nome relativo di un oggetto è sempre specificato nella forma `tipo=valore`, dove `tipo` è la stringa identificativa dell'attributo usato per indicare l'oggetto (come `dc`, `ou` o `cn`) mentre `valore` ne indica il contenuto, pertanto si potrà avere qualcosa come `ou=Ricerca`, o `cn=Simone Piccardi`. Per specificare un *Distinguished Name* basta scrivere la lista dei nomi dei singoli oggetti separati da virgole, iniziando dal fondo dell'albero fino e risalendo fino alla radice; ad esempio nel caso di fig. 1.2 per indicare l'oggetto di tipo *Person* corrispondente a Simone Piccardi si userà:

```
cn=Simone Piccardi,ou=ricerca,o=Truelite Srl,dc=truelite,dc=it
```

Tutte le volte che si installa[12] un server LDAP occorre scegliere quale delle due strutturazioni illustrate si vuole usare, e definire in quale punto dell'albero (all'interno di uno dei due schemi generali possibili) compariranno i dati in esso presenti. Questo definisce quello che sarà il *suffisso*[13] per tutti i nomi degli oggetti contenuti nell'albero; così se si è scelto il criterio basato sui nomi a dominio potremo avere un suffisso del tipo `dc=truelite,dc=it`, mentre se si è scelto il criterio geografico potremo avere un suffisso del tipo `o=Truelite Srl,c=IT`.

1.1.3 I file in formato LDIF

In generale ogni server LDAP mantiene i dati contenuti nell'elenco tramite un opportuno meccanismo di supporto (per OpenLDAP tratteremo l'argomento in sez. 1.2.3). Per permettere di trasferire i dati fra server diversi è stato definito un apposito formato di interscambio, l'*LDAP Data Interchange Format* (in breve LDIF) che permette di esprimere una qualunque voce usando dei semplici file di testo.

Lo standard che definisce il formato LDIF è specificato dall'RFC 2849, questo prevede che ogni voce nel database possa essere espressa in testo semplice, con una schematizzazione della stessa nella forma:

```
# commento
dn: <distinguished name>
<nome attributo>: <valore>
<nome attributo>: <valore>
...
```

Al solito il carattere "`#`" viene usato per indicare una linea di commento e le righe vuote vengono ignorate. Ciascuna voce deve essere introdotta da una riga iniziante per `dn:` che ne dichiari il *Distinguished Name* in modo da identificarla univocamente; questo verrà espresso come visto in sez. 1.1.2.

Alla dichiarazione del *Distinguished Name* seguono le assegnazioni dei vari attributi nella forma illustrata, questi devono essere indicati tramite il relativo nome (come `cn` o `objectClass`), seguito dal carattere "`:`" da uno spazio e poi dal valore. Per poter utilizzare uno specifico attributo occorrerà ovviamente che l'oggetto derivi da una *objectClass* che lo definisce; se il file specifica un oggetto che deve essere creato si può indicare da quali *objectClass* derivarlo attraverso l'attributo speciale `objectClass`.

Se una riga è troppo lunga può essere fatta proseguire sulla successiva usando come primo carattere di quest'ultima o uno spazio o un tabulatore.[14] Si tenga inoltre conto che gli ulteriori

[12] se usate Debian, almeno a partire da *Sarge*, la scelta è già stata fatta dal sistema del *debconf*, che utilizza solo i nomi a dominio.

[13] in seguito faremo spesso riferimento ad esso anche con il nome di *base*, in quanto questo suffisso viene sempre usato, quando si esegue una ricerca, come base della stessa.

[14] questo significa che la specificazione del nome di un attributo deve sempre essere iniziata sul primo carattere di una riga.

spazi iniziali che seguono il carattere ":" non vengono ignorati, e che la presenza di spazi multipli all'interno dei valori viene mantenuta tale e quale; pertanto se non si vogliono spazi aggiuntivi occorre stare attenti a non metterceli.

Qualora il valore dell'attributo non sia esprimibile con caratteri stampabili, o inizi per uno spazio o con i caratteri riservati ":" e "<" questo dovrà essere specificato usando una sintassi diversa; le opzioni sono due, la prima prevede l'uso di un valore codificato in formato *Base-64*[15], la seconda invece prevede la lettura dei dati direttamente da una fonte di dati esterna specificata tramite la sua URL. In tal caso la dichiarazione del valore di un attributo userà le due forme alternative "::" e ":<" ad esempio per specificare il valore di un attributo `jpegPhoto` si potranno usare le sintassi:

```
jpegPhoto:: /9j/4AAQSkZJRgABAgAAZABkAAD/7AARRHVja3kAAQAEAAAAUAAA/+4ADkFk
 b2J1AGTAAAAAAf/bAIQAAgICAgICAgICAgMCAgIDBAMCAgMEBQQEBAQEBQYF
 BQUFBQUGBgcHCAcHBgkJCgoJCQwMDAwMDAwMDAwMDAwMDAEDAwMFBAUJBgYJ
 ...
jpegPhoto:< file:///path/to/photo.jpeg
```

Qualora un attributo compaia più volte questo dovrà essere ripetuto su righe distinte; infine si possono inserire più voci all'interno di uno stesso file separandole con una o più righe vuote. Un esempio di file LDIF è allora il seguente:

```
# Truelite Srl, Contacts, truelite.it
dn: cn=Truelite Srl,ou=Contacts,dc=truelite,dc=it
cn: Truelite Srl
sn: Srl
mail: info@truelite.it
telephoneNumber: 0557879597
facsimileTelephoneNumber: 0557333336
postalAddress:: VmlhIE1vbmZlcnJhdG8sIDYKRmlyZW56ZSwgRkkgNTAxNDIKSXRhbHk=
labeledURI: http://www.truelite.it
o: Truelite Srl

# Antonio Javier Russo, Contacts, truelite.it
dn: cn=Antonio Javier Russo,ou=Contacts,dc=truelite,dc=it
cn: Antonio Javier Russo
sn: Russo
mobile: 0471XXXXXX
fileAs: Russo, Antonio
mail: russo@indirizzo.fake.it
mail: antonio.russo@no.spamming.allowed.de
o: Associazione Software Libero
title: Coordinatore
```

in cui sono definite due voci nella unità operativa `Contacts` (nel caso utilizzata per mantenere un indirizzario); si noti come trattandosi di voci relative a tipi di dati diversi queste siano realizzate con oggetti diversi, che hanno diversi attributi.

1.1.4 Il meccanismo delle interrogazioni su LDAP

Come accennato in sez. 1.1.1 LDAP nasce come un sistema in grado di offrire dei servizi di elenco a livello globale. Questo significa che, come accade con il meccanismo delle delegazioni del DNS, è possibile suddividere l'albero generico illustrato in sez. 1.1.2 (che si usi la strutturazione geografica o quella dei domini) in maniera che ciascun server possa rispondere per la sua sezione di albero, ottenendo così una distribuzione delle informazioni.

[15] questa è una semplice codifica, le cui specifiche si possono trovare all'interno dell'RFC 2045, che permette di esprimere un qualunque dato binario codificando lo stesso in un formato che prevede l'uso di 63 caratteri stampabili (le lettere maiuscole e minuscole, i numeri ed i caratteri "+" e "/").

Per poter richiedere le informazioni il protocollo LDAP prevede l'utilizzo di una URL estesa che consenta di eseguire in maniera generale una richiesta ad un server LDAP;[16] questa URL ha una forma generica del tipo:

```
ldap://server.dominio.it/base?attributi?profondità?filtro
```

dove la prima parte, che specifica l'indirizzo del server da contattare,[17] è identica a quella di una URL consueta,[18] mentre la seconda parte, che indica la ricerca da effettuare, ha una sua sintassi specifica composta, come mostrato nell'esempio, da diversi elementi separati dal carattere "`?`".

Il primo elemento è l'indicazione del punto di partenza della ricerca all'interno dell'albero dei dati, la cosiddetta *base* della ricerca;[19] questa deve essere espressa con il suo *Distinguished Name*. Il secondo elemento indica la lista, separata da virgole, dei nomi degli attributi che si intendono cercare, se non specificato verranno restituiti tutti gli attributi presenti. Il terzo elemento è una parola chiave che indica la *profondità* della ricerca (vedi tab. 1.17) e l'ultimo un *filtro di ricerca* (tratteremo il significato di questi due ultimi elementi in sez. 1.2.5).

Inoltre dato che esistono diverse modalità con cui si può contattare un server (torneremo su questo in sez. 1.2.2 e sez. 1.2.3), l'identificativo iniziale della URI, oltre a quanto mostrato nell'esempio precedente, può assumere una delle tre forme illustrate in tab. 1.1, che identificano la modalità con cui collegarsi al server.

Indicatore	Significato
`ldap://`	indica una normale connessione tramite socket.
`ldaps://`	indica una connessione cifrata con SSL (vedi sez. 1.3.2)
`ldapi://`	indica una connessione ad un socket locale.

Tabella 1.1: Gli indicatori speciali per le URL che identificano il servizio LDAP.

Il caso più comune di uso di interrogazione col protocollo LDAP resta quello in cui un client effettua una richiesta ad un server via rete,[20] usando una URL nella forma appena descritta, e ottiene direttamente da questo la risposta, secondo lo schema illustrato in fig. 1.3. Questo caso ovviamente comporta che il server gestisca i dati relativi alla richiesta; quando si usano delle risorse *locali* questo avviene sempre, e nella nostra analogia con il DNS ciò equivale all'effettuare richieste solo per il dominio di cui siamo direttamente responsabili.

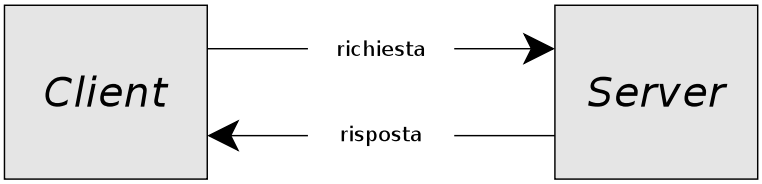


Figura 1.3: Struttura di una richiesta diretta ad un server LDAP.

Dato che molto spesso LDAP viene utilizzato solo per mantenere informazioni locali (come una rubrica di indirizzi condivisa) quello di fig. 1.3 è lo scenario di utilizzo più comune, ma benché ciò sia meno diffuso è possibile uscire dall'ambito locale ed eseguire delle interrogazioni generiche in ambito globale. In questo caso il meccanismo di funzionamento del protocollo è simile a quello del DNS, è necessario cioè inserire il nostro server all'interno di una gerarchia, così che sia possibile redirigere le richieste verso server di livello superiore che possono a loro volta redirigere verso altri server subordinati che mantengano le informazioni che cerchiamo.

[16] così come con una URL usuale è possibile interrogare un server web.

[17] e anche una eventuale porta, se si usa qualcosa come `ldap://server:porta`.

[18] a parte l'uso di `ldap` al posto di `http`.

[19] in genere corrisponde al *suffisso* della radice dell'albero illustrato in sez. 1.1.2, ma le ricerche possono essere effettuate a partire da qualunque profondità e non solo dalla radice.

[20] che si usi o meno SSL dal punto di vista del protocollo è indifferente.

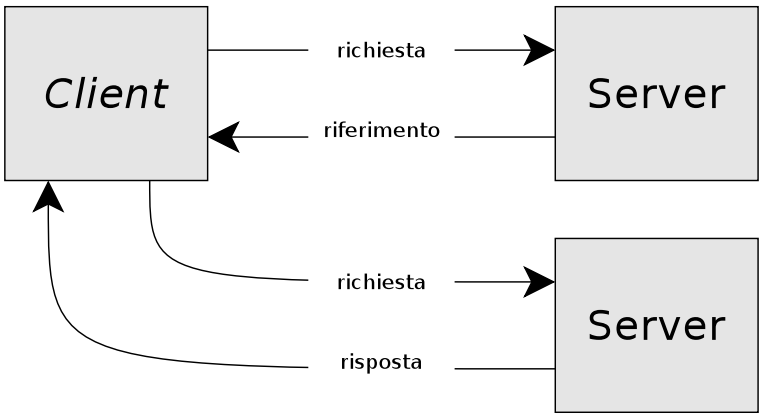


Figura 1.4: Il meccanismo dei *referral* di LDAP.

La redirezione avviene tramite il meccanismo detto dei *referral*, che si è illustrato in fig. 1.4. Si può cioè configurare un server LDAP che fornisce dati per un certo dominio perché per richieste che escono da detto dominio possa fornire un *riferimento* ad un server di livello superiore in grado di rispondere (o di fornire riferimenti ad altri server). Allo stesso modo un server di livello superiore può redirigere richieste ad altri server sotto di lui che mantengono quella particolare sezione di albero.

La potenza del meccanismo dei *referral* è che, come per il DNS, esso è del tutto trasparente rispetto alle richieste di un client, che riceverà solo la risposta finale. Ovviamente perché tutto questo sia possibile occorre, come per il DNS, la presenza di una gerarchia globale di server. Benché questa funzionalità sia scarsamente utilizzata,[21] una tale gerarchia è possibile,[22] e ad esempio lo stesso progetto della *OpenLDAP Foundation* gestisce un servizio di questo tipo su `ldap://root.openldap.org/`.

Si tenga presente comunque che anche quando non si è interessati a far parte di un servizio strutturato a livello globale, questa caratteristica di LDAP risulta particolarmente utile in quanto consente, anche all'interno di una singola organizzazione, di suddividere le informazioni (ad esempio relative alle varie divisioni) su server diversi messi in relazione fra loro e mantenuti coerenti in un unico albero grazie alla presenza dei riferimenti.

1.2 Installazione e configurazione di base di *OpenLDAP*

Tratteremo in questa sezione l'installazione e la configurazione di base di *OpenLDAP*, un pacchetto composto da un server LDAP e da un insieme di librerie e programmi utente per l'accesso al servizio fornito dal server. Il pacchetto è sviluppato dalla *OpenLDAP Foundation* e viene usato come implementazione di riferimento di LDAP in tutte le distribuzioni di GNU/Linux. Siccome alcune opzioni di configurazione sono state modificate durante lo sviluppo, se non specificato altrimenti si farà riferimento ai valori utilizzati nella versione 2.2.

1.2.1 Installazione ed avvio del servizio

Tutte le più importanti distribuzioni forniscono versioni già pacchettizzate di OpenLDAP, per cui in generale tutto quello che serve è usare il rispettivo gestore di pacchetti per eseguire l'installazione. In certi casi però la versione fornita non ha tutti i supporti necessari (era ad

[21] l'uso prevalente di LDAP è per mantenere dati ed informazioni relativi alla propria organizzazione.
[22] che però, per le scarse ragioni che ne motivino un utilizzo a livello globale, non si è mai affermata.

esempio il caso di quella fornita con la versione *woody*[23] di Debian, che mancava del supporto per SSL/TLS) o non è sufficientemente aggiornata e può essere necessario reinstallare dai sorgenti.

I sorgenti di OpenLDAP sono disponibili sul sito del progetto a partire dalla pagina web `http://www.openldap.org/software/download/` e sono mantenuti in due serie diverse; la prima serie, marcata *release*, viene rilasciata ogni volta che sono disponibili nuove funzionalità e correzioni; quando una di queste *release* si è dimostrata, nel successivo uso, sufficientemente stabile e priva di errori, viene marcata *stable*.

La scelta fra le due serie dipende essenzialmente dalle proprie necessità, è comunque da preferire la versione *stable* fintanto che non sono strettamente necessarie le funzionalità presenti in una *release*; i file sono sempre nella forma `openldap-VERSIONE.tgz`, che vanno scompattati al solito nella directory in cui si eseguirà la compilazione.

Molte delle funzionalità di OpenLDAP si basano su altri pacchetti e librerie, per poterle utilizzare è ovviamente necessario che questi siano presenti; non è fondamentale che siano presenti tutti (lo script di configurazione consente di selezionare quelli che servono) ma ovviamente per quelli che si intende utilizzare dovranno essere presenti anche i relativi pacchetti di sviluppo. L'elenco dei prerequisiti è il seguente, torneremo sulla configurazione di alcune di queste funzionalità in sez. 1.3:

SSL/TLS Per l'uso di connessioni cifrate con il *Transport Layer Security*[24] OpenLDAP richiede la presenza delle librerie OpenSSL; non si potrà avere una piena conformità allo standard LDAPv3 se OpenLDAP non rileva la presenza di OpenSSL.

Kerberos OpenLDAP supporta i servizi di autenticazione basati su Kerberos, in particolare si possono utilizzare le due implementazioni libere di Kerberos V: Heimdal e MIT Kerberos.

SASL OpenLDAP può utilizzare i servizi della libreria SASL (*Simple Authentication and Security Layer*) per l'autenticazione degli utenti. Questo richiede la presenza delle librerie Cyrus SASL senza le quali non si potrà avere una piena conformità allo standard LDAPv3.

DBM per mantenere i suoi dati un server OpenLDAP necessita del supporto di un database su cui salvarli,[25] questo può essere sia Berkley DB (la versione 4.2 distribuita da `http://www.sleepycat.com/`) che un qualunque altro database che supporti l'interfaccia generica DBM di BSD (come GDBM, disponibile su `http://www.gnu.org/software/gdbm/`) che un qualunque altro tipo di supporto (che può essere realizzato, come elencato in tab. 1.3, anche tramite script di shell).

TCP wrapper OpenLDAP supporta i *TCP wrapper* per un controllo di accesso elementare degli accessi sulla base degli indirizzi di provenienza delle connessioni al server, che può essere utilizzato qualora essi siano stati installati.

La procedura di installazione dei sorgenti è molto simile a quella standardizzata dagli *autotools GNU*,[26] e viene controllata dallo script `configure`; questo prevede una serie di variabili di ambiente, riportate in tab. 1.2, che permettono di modificare il compilatore e le relative opzioni.

[23] versione stabile al tempo in cui si è iniziato a scrivere queste dispense.

[24] questa è la standardizzazione ufficiale fatta dalla IETF del protocollo di cifratura delle comunicazioni a livello di socket introdotta con il *Secure Socket Layer* di Netscape.

[25] si tenga presente che non si tratta di avere un database relazionale, basta un database in grado di mantenere archiviati i dati eseguendo su di essi ricerche veloci.

[26] per una descrizione più dettagliata di questa procedura si può fare riferimento a [AGL], sez.4.2.1.

In generale lo script è in grado di riconoscere automaticamente i valori ottimali per le suddette variabili, e non è necessario utilizzarle.

Variabile	Utilizzo
`CC`	specifica un altro compilatore.
`CFLAGS`	specifica opzioni di compilazione addizionali.
`CPPFLAGS`	specifica opzioni addizionali per il preprocessore.
`LDFLAGS`	specifica opzioni per il linker.
`LIBS`	specifica librerie addizionali.

Tabella 1.2: Le variabili di ambiente usate dallo script di configurazione per la compilazione di OpenLDAP.

Lo script di configurazione riconosce le opzioni standard degli autotools, e permette di abilitare le varie funzionalità aggiuntive di OpenLDAP con le opzioni `--enable-FEATURE` e `--with-PACKAGE`;[27] un elenco di tutte le opzioni disponibili si può ottenere invocando lo script di configurazione come:

```
./configure --help
```

Opzione	Default	Significato
`--enable-debug`	si	abilita il supporto per il debug.
`--enable-syslog`	auto	abilita il supporto per l'uso del *syslog*.
`--enable-ipv6`	auto	abilita il supporto per IPv6.
`--enable-local`	auto	abilita il supporto per i socket locali.
`--enable-cleartext`	si	abilita il supporto per mantenere le password in chiaro.
`--enable-crypt`	no	abilita il supporto per mantenere le password cifrate con `crypt`.
`--enable-wrappers`	no	abilita l'uso dei *TCP wrapper*.
`--enable-bdb`	si	abilita il supporto dei dati su Berkley DB 4.2.
`--enable-ldbm`	no	abilita il supporto dei dati sull'interfaccia generica DBM.
`--enable-perl`	no	abilita il supporto dei dati tramite script perl.
`--enable-shell`	no	abilita il supporto dei dati tramite script di shell.
`--enable-sql`	no	abilita il supporto dei dati su un database relazionale.
`--enable-slurpd`	auto	abilita la compilazione del server di replicazione (vedi sez. 1.4).
`--with-cyrus-sasl`	auto	abilita l'utilizzo delle librerie SASL di Cyrus.
`--with-threads`	auto	abilita l'utilizzo dei thread.
`--with-tls`	auto	abilita l'utilizzo del TLS con OpenSSL.

Tabella 1.3: Le opzioni usate dallo script di configurazione per la compilazione di OpenLDAP.

In tab. 1.3 si sono riportate le opzioni principali ed il relativo valore di default, qualora si voglia abilitare una funzionalità che usualmente non è attiva (ad esempio i TCP wrapper) la si dovrà abilitare esplicitamente con:

```
./configure --enable-wrappers
```

Una volta eseguito lo script di configurazione prima si dovranno costruire le dipendenze con `make depend`, poi compilare con `make`. A questo punto è anche disponibile il comando `make`

[27] dove `FEATURE` e `PACKAGE` indicano rispettivamente una funzionalità da abilitare o un pacchetto esterno da utilizzare.

`test` per verificare il funzionamento del pacchetto. Infine `make install` (ovviamente dato dall'amministratore) installerà sia il server, che le librerie, che i programmi di utilità a riga di comando.[28]

Se invece si procede all'installazione dalla versione fornita dalla propria distribuzione si tenga presente che in alcuni casi le varie parti di OpenLDAP sono separate in pacchetti diversi; ad esempio nel caso di Debian il server è nel pacchetto `slapd` (dato che questo è il nome del demone che fornisce il servizio), le librerie sono invece nel pacchetto `libldap2` e i programmi di utilità a riga di comando in `ldap-utils`.[29]

Qualora si sia installato OpenLDAP dai sorgenti si troverà un file di configurazione di esempio per il server in `/usr/local/etc/openldap`; in genere i pacchetti delle distribuzioni installano i file di configurazione nella directory `/etc/ldap`, in questa directory si trova il file di configurazione delle librerie (per Debian è `ldap.conf`), vari file di configurazione dei programmi di utilità ed il file di configurazione del server (per Debian è `slapd.conf`).

Su una macchina che deve fare da server LDAP, benché non siano strettamente necessarie, si installano tutte le componenti. Per utilizzare OpenLDAP a livello client invece l'unica componente necessaria è la libreria `libldap` che viene usata da tutti i programmi (comprese le utilità fornite da OpenLDAP stesso) che devono interrogare un server LDAP; in genere si installano anche i programmi di utilità.

Nel caso si usi Debian installando il server *debconf* si curerà di chiedere anche alcuni dei parametri di base per la creazione di una struttura minimale per il proprio albero dei dati (in sostanza le informazioni necessarie a stabilire quale sarà la propria *radice locale*), ed una password da associare ad un utente amministrativo che avrà accesso completo al database. Installando il client *debconf* chiede invece i dati del server e della radice da usare come valore preimpostato.[30]

Una volta installato il server questo potrà essere avviato direttamente con il comando `slapd`. Questo, se eseguito senza l'opzione `-d` che abilita la modalità di debug, crea un nuovo processo figlio e si dissocia dal terminale continuando a girare come demone, utilizzando le impostazioni che trova nel file di configurazione `/etc/ldap/slapd.conf` (che tratteremo in sez. 1.2.3). Un file di configurazione alternativo può essere indicato con l'opzione `-f`.

Di default il server si pone in ascolto per connessioni TCP in chiaro su tutte le interfacce di rete usando la porta standard del servizio LDAP (la 389); si può però modificare questo comportamento con l'opzione `-h` che prende come parametro una URL con gli indicatori di tab. 1.1, che definisce le modalità (e l'indirizzo locale) con cui ci si potrà connettere al server. Come riportato in tab. 1.1, oltre la modalità classica si ha a disposizione (se si sono abilitati i relativi supporti) la possibilità di usare connessioni cifrate con SSL/TLS e la connessione su socket locali. Nei primi due casi si dovranno specificare l'indirizzo IP e la porta nella sintassi classica delle URL,[31] mentre nel terzo si dovrà specificare il pathname assoluto del socket.[32]

Si sono riportate le altre opzioni principali di `slapd` in tab. 1.4. Al solito per l'elenco completo si può fare riferimento alla pagina di manuale del comando; inoltre buona parte di esse può essere specificata con apposite direttive nel file di configurazione (vedi sez. 1.2.3). Normalmente per lanciare il server è sempre opportuno utilizzare l'opportuno script di avvio messo a disposizione dalla propria distribuzione (nel caso di Debian `/etc/init.d/slapd`). Nel caso di Debian

[28] al solito il default è l'installazione sotto `/usr/local/bin`.

[29] a questi si aggiungono poi, qualora si debbano compilare altri programmi che li richiedono, i pacchetti di sviluppo.

[30] le ultime versioni del pacchetto cercano di automatizzare al massimo il procedimento usando il dominio della macchina specificato in fase di installazione.

[31] cioè qualcosa nella forma "`ldap://indirizzo:porta/`" per ascoltare in chiaro sulla porta specificata, o "`ldaps:///`" per ascoltare su SSL sull'indirizzo generico e sulla porta standard (che per LDAP su SSL è la 636); la forma completa vista in sez. 1.1.4 in questo caso non si applica.

[32] si ricordi che i socket locali sono un meccanismo di intercomunicazione fra processi basato sull'uso di un file speciale, di tipo socket; per maggiori dettagli si veda la sez. 1.2 di [AGL].

Indicatore	Significato
`-d`	indica il livello di debug, secondo i valori di tab. 1.5.
`-s`	indica la priorità da assegnare ai messaggi di `slapd` nel sistema del *syslog*.
`-l`	indica quale facility usare nel sistema del *syslog*.
`-f`	indica un file di configurazione alternativo.
`-h`	indica a quali collegamenti risponde il server.
`-u`	indica l'utente per conto del quale eseguire il server.
`-g`	indica il gruppo per conto del quale eseguire il server.
`-t`	esegue un controllo sintattico del file di configurazione.

Tabella 1.4: Opzioni del comando `slapd`.

poi in `/etc/defaults/slapd` sono utilizzabili le due variabili di ambiente `SLAPD_SERVICES` e `SLAPD_OPTIONS` da utilizzare rispettivamente per passare agli script di avvio del servizio una diversa URL rispetto al default di `ldap:///`,[33] e per indicare eventuali opzioni aggiuntive (ad esempio di debug) da usare all'avvio del server.

Una delle opzioni più importanti è `-d` che attiva la modalità di debug; essa permette di eseguire il server interattivamente, ottenendo così i messaggi direttamente sul terminale. L'opzione richiede un parametro che indica quali informazioni di debug debbano essere stampate, questo viene interpretato come maschera binaria, ed ogni bit attiva la stampa di una certa classe di messaggi, secondo lo schema di tab. 1.5.

Indicatore	Sigla	Significato
0	–	nessuna informazione.
1	`trace`	traccia le chiamate alle singole funzioni.
2	`packet`	informazioni sulla gestione dei pacchetti.
4	`args`	informazioni di debug generico.
8	`conns`	informazioni sulla gestione della connessione.
16	`BER`	stampa i pacchetti ricevuti ed inviati.
32	`filter`	informazioni sull'uso dei filtri.
64	`config`	informazioni sulla scansione della configurazione.
128	`ACL`	informazioni sull'uso delle regole di accesso.
256	`stats`	statistiche delle operazioni e delle connessioni.
512	`stats2`	statistiche sulle voci inviate.
1024	`shell`	stampa la comunicazioni con il *backend* di shell (vedi sez. 1.2.3).
2048	`parse`	informazioni sulla scansione delle voci.
4096	`cache`	informazioni sulla cache (inutilizzate).
8192	`index`	informazioni sugli indici (inutilizzate).
16384	`sync`	informazioni sulla replicazione LDAPSync.
-1		abilita tutte le informazioni.

Tabella 1.5: Significato dei bit di debug.

Specificando un valore nullo si disattivano i messaggi di debug, mentre con -1 si attivano tutti quanti; in genere per non restare "*affogati*" nell'output del server è opportuno selezionare solo i messaggi della classe cui fa riferimento il problema che si vuole investigare; questo si fa semplicemente sommando i valori riportati nella prima colonna di tab. 1.5. Così se si vogliono verificare le modalità della connessione su SSL/TLS e la verifica delle regole di accesso si potrà invocare il programma come:

```
slapd -d 8 -h ldaps:///
```

[33] la variabile deve contenere il valore del parametro per l'opzione `-h`.

1.2.2 La configurazione di base del client

La configurazione dei servizi LDAP coinvolge sia il lato server che quello client. Inizieremo dalla configurazione del lato client, che in realtà non è altro che la configurazione delle librerie di accesso che vengono usate da quasi tutti i programmi di interrogazione al servizio. Questo significa che (a meno che non sia stato implementato in maniera indipendente un accesso diretto) qualunque programma che si rivolge ad un server LDAP risentirà di dette impostazioni.

Come accennato il file di configurazione generale delle librerie di accesso è `ldap.conf`; questo contiene una serie di direttive, una per riga, nella forma classica di una parola chiave seguita da un valore, separati da uno o più spazi (o altri caratteri vuoti). Quando un valore contiene degli spazi questo deve essere indicato delimitandolo con delle virgolette, se queste ultime sono contenute in un valore devono essere protette dal carattere di escape "\" che può essere usato anch'esso nello stesso modo. Come al solito le righe che iniziano per un "`#`" sono considerate commenti e vengono ignorate come le righe vuote,[34] infine una riga che inizia con uno spazio è considerata la continuazione della precedente.

Direttiva	Significato
`BASE`	indica il *Distinguished Name* usato come base per le ricerche nell'albero.
`URI`	indica la URL (nel formato illustrato in tab. 1.1) con cui contattare il server.
`BINDDN`	indica il *Distinguished Name* dell'utente per conto del quale viene eseguito il collegamento di default (se non lo si indica il collegamento è anonimo).
`BINDPW`	la password da usare per collegarsi con il *Distinguished Name* specificato dalla precedente `BINDDN`.
`HOST`	indica il nome o l'indirizzo IP del server a cui collegarsi di default, nella forma `indirizzo:porta`.
`PORT`	indica la porta da usare quando ci si collega ad un server LDAP.
`SIZELIMIT`	indica un limite massimo di dimensioni nelle ricerche, deve essere un valore positivo; un valore nullo (il default) indica una dimensione illimitata.
`TIMELIMIT`	indica un limite massimo di tempo nell'eseguire una ricerca, deve essere un valore positivo; un valore nullo (il default) indica un tempo illimitato.
`ROOTBINDDN`	specifica l'utente del database con il quale devono essere eseguiti gli accessi a LDAP effettuati da un processo che viene eseguito con i privilegi di amministratore (necessita della relativa password nel file `/etc/ldap.secret`).
`SCOPE`	indica la profondità su cui viene eseguita la ricerca dei dati, prende i valori illustrati in tab. 1.17.
`DEBUG`	indica il livello di debug per i messaggi di errore del client, prende un valore numerico con lo stesso significato visto in tab.1.5 per l'analoga direttiva del server.

Tabella 1.6: Le principali direttive di `ldap.conf`.

L'impostazione principale che si esegue in questo file è quella del server di default, a cui verranno rivolte le interrogazioni quando non se ne è indicato uno esplicitamente. Questo può essere fatto con le direttive `HOST` e `PORT`, cui deve seguire rispettivamente l'indirizzo IP (o il nome) del server e il numero di porta; quest'ultimo può anche essere indicato direttamente nell'argomento di `HOST` scrivendolo separato con un carattere "`:`" di seguito al nome (o all'indirizzo); si può anche non specificare la porta, nel qual caso sarà utilizzata la porta standard associata al servizio `ldap` (la 389).

[34] stando attenti a che le righe siano effettivamente vuote: infatti la presenza di uno spazio all'inizio di una riga implica che questa è la continuazione della precedente, e si possono avere effetti inaspettati quando una riga *sembra* essere vuota ma non lo è davvero.

Le direttive `HOST` e `PORT` sono comunque deprecate in favore della nuova direttiva `URI`; questa prende un indirizzo nella forma della URL estesa già illustrata in tab. 1.1, che specifica l'indirizzo del server a cui ci si rivolge, o, nel caso di `ldapi://`, il pathname del socket locale su cui è in ascolto il server.

Un'altra direttiva di uso comune è `BASE` che permette di definire il *Distinguished Name* da usare come suffisso base per le ricerche; serve cioè ad impostare la radice dell'albero dei dati presenti sul server, ma può essere anche usato per restringere l'accesso delle interrogazioni ad una sezione particolare dell'albero (ad esempio alla sola parte che contiene l'indirizzario, o a quella relativa ad una sola unità operativa).

Le altre direttive comuni più usate sono riportate in tab. 1.6,[35] il file comunque prevede la presenza di ulteriori direttive relative alla configurazione di alcune funzionalità specifiche come SASL o SSL che non sono riportate qui. Le esamineremo più avanti quando tratteremo i relativi argomenti, per l'elenco completo si può comunque consultare la pagina di manuale, accessibile con `man ldap.conf`.

Oltre alla configurazione generale di `ldap.conf` un utente può creare una sua configurazione personalizzata usando le stesse direttive all'interno di un file `.ldaprc` posto nella sua home directory; così facendo le nuove impostazioni soprassiederanno quelle generali. Una ulteriore capacità di personalizzazione si può ottenere con l'uso di alcune variabili di ambiente, ad esempio con `LDAPCONF` si può indicare un file di configurazione alternativo. Un elenco di queste variabili è riportato in tab. 1.7.

Variabile	Significato
`LDAPNOINIT`	se definita disabilita tutti i valori di default.
`LDAPCONF`	deve essere assegnata al nome del file da usare come file di configurazione al posto di `ldap.conf`.
`LDAPRC`	indica il nome del sostituto di `.ldaprc` che verrà usato per le configurazioni utente, da cercare nella directory corrente o nella home.
`LDAP<XXX>`	soprassiede il valore della direttiva `XXX` del file di configurazione (ad esempio `LDAPURI` soprassiede il valore di `URI`).

Tabella 1.7: Variabili di ambiente che controllano il comportamento delle librerie di accesso di OpenLDAP.

In generale tutto quello che c'è da impostare sul lato client è il server a cui rivolgersi e la base della ricerca; il contenuto tipico di un file `ldap.conf` è pertanto analogo al seguente, estratto dal file installato su Debian Sid (nel caso è *debconf* che si cura di chiedere l'indirizzo del server):

```
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

#BASE   dc=example, dc=com
#URI    ldap://ldap.example.com ldap://ldap-master.example.com:666
BASE    dc=truelite, dc=it
URI     ldaps://ldap.truelite.it

#SIZELIMIT      12
#TIMELIMIT      15
#DEREF          never
```

[35] i nomi delle direttive in tabella sono indicati in minuscolo, ma il loro valore è *case insensitive* e negli esempio sono spesso utilizzate, in maniera altrettanto valida, in caratteri maiuscoli.

1.2.3 La configurazione di base del server

La configurazione del server è ovviamente molto più complessa e ne tratteremo in questa sezione solo gli aspetti generali, rimandando a sez. 1.3 i dettagli relativi alle funzionalità più avanzate. Come accennato il file di configurazione di default è `slapd.conf`, ed ha un formato identico a quello di `ldap.conf`.

Il file prevede tre tipi di direttive di configurazione: quelle generali che sono applicabili a tutto il server, quelle specifiche che sono applicabili solo ad un tipo di supporto e quelle ancor più specifiche che si applicano ad un singolo database. Una delle caratteristiche più interessanti di OpenLDAP infatti è che una singola istanza del server `slapd` può gestire diversi database, inoltre per mantenere i dati è possibile usare diversi supporti, (quelli che in gergo vengono chiamati *backend*) che sono disponibili in forma modulare.

La possibilità di avere diversi *backend* consente così di svincolare completamente la parte del server che gestisce le operazioni previste dal protocollo LDAP, dal codice che gestisce i dati di un database, garantendo una enorme flessibilità nel mantenimento degli stessi sui supporti più vari.[36]

Due direttive speciali, `backend` e `database`, servono a definire l'inizio di una sezione del file di configurazione in cui si immettono rispettivamente le direttive di configurazione relative ad uno specifico supporto (volte a controllarne le caratteristiche specifiche) e ad uno specifico database (coi dati ad esso attinenti); la fine di una sezione è delimitata dall'inizio di un'altra sezione con la presenza di una di queste due direttive, o dalla fine del file di configurazione.

Le direttive inoltre prevedono una gerarchia ed un ambito di validità. Quelle globali possono essere usate anche all'interno di una sezione `backend` o `database` ma in tal caso il loro valore sarà applicato soltanto a detta sezione, e lo stesso vale per direttive relative ad un supporto usate in una sezione `database`. Inoltre una direttiva utilizzata in una sezione `backend` soprassiederà il valore assegnato globalmente, ed una direttiva utilizzata in una sezione `database` soprassiederà sia il valore globale che quello assegnato in una sezione `backend`. Se una direttiva compare più volte all'interno di una stessa sezione verrà utilizzato il valore dell'ultima assegnazione. Questo in sostanza comporta una struttura del tipo:

```
# Configurazioni di carattere generale
<direttiva generale> <valore>
<direttiva generale> <valore>
...
# Primo tipo di supporto
backend <tipo1>
<direttiva specifica per supporto> <valore>
...
database <tipo1>
<direttiva specifica per database> <valore>
...
database <tipo1>
<direttiva specifica per database> <valore>
...
# Secondo tipo di supporto
backend <tipo2>
<direttiva specifica per supporto> <valore>
...
database <tipo2>
```

[36] è ad esempio con uno di questi supporti che si può gestire la cosiddetta *proxy-replication* che tratteremo in sez. 1.4.4.

```
<direttiva specifica per database> <valore>
...
```

in cui prima si dichiara il tipo di supporto, seguito dalle configurazioni specifiche dello stesso, e poi i vari database che utilizzano quel supporto (con le loro configurazioni specifiche).

Tipo	Significato
`bdb`	usa come *backend* Berkley DB 4.2 (vecchia interfaccia).
`hdb`	nuova interfaccia (evoluzione di `bdb`) per Berkley DB .
`ldbm`	usa un qualunque database compatibile con l'interfaccia DBM.
`ldap`	fa da proxy verso un altro server LDAP.
`meta`	fa da proxy verso un gruppo di server LDAP.
`sql`	mappa i dati in un database relazionale su LDAP.
`relay`	fa da proxy nei confronti di un database locale (sperimentale).
`shell`	ottiene i dati da script di shell.
`perl`	ottiene i dati da script Perl.
`monitor`	riporta dati statistici del server.
`passwd`	riporta i dati di `/etc/passwd`.
`dnssrv`	riporta i dati da un DNS.
`null`	non fa nulla.

Tabella 1.8: I vari tipi di supporto per i dati di `slapd`.

Entrambe le direttive `backend` e `database` prendono come argomento uno degli identificativi dei vari supporti disponibili riportati in tab. 1.9.[37] Il supporto più utilizzato fino all'incirca alla versione 2.3 era `bdb`, che indica l'utilizzo del Berkley DB 4.2 (si ricordi quanto detto in sez. 1.2), che è quello consigliato dal progetto. L'uso di `bdb` permette di utilizzare le funzionalità avanzate di Berkley DB, che supporta le transazioni[38] e consente l'accesso simultaneo in lettura e scrittura da più processi. Nelle versioni più recenti il viene invece consigliato l'uso del nuovo modulo `hdb`,[39] che si appoggia sempre a Berkley DB ed ha le stesse caratteristiche di `bdb`, utilizzando però una interfaccia più efficiente.

L'unico altro supporto che è stato effettivamente utilizzato in ambiti di produzione, ma ormai sostanzialmente abbandonato, è `ldbm` che implementa l'interfaccia generica di DBM, (la libreria di funzioni per gestire un database su file all'interno di qualunque programma) da cui deriva anche Berkley DB. L'interfaccia non supporta le funzionalità avanzate, e non sono disponibili le transazioni; questo comporta che mentre è possibile l'accesso in lettura da più processi, quello in scrittura deve essere esclusivo.[40]

Tutti i supporti disponibili sono elencati in tab. 1.9. In generale i supporti si possono suddividere in tre categorie generali: quelli che mantengono effettivamente dei dati, quelli che servono come proxy per dati mantenuti altrove, e quelli che generano i dati al volo. Il primo gruppo (quello usato normalmente) contiene i precedenti `bdb` e `ldbm` e `hdb`, che deriva da `bdb` con alcune ottimizzazioni sulla gestione degli indici, che sono mantenuti in memoria e non su file (ottenendo maggiori prestazioni in scrittura, ma un avvio più lento).

Al secondo gruppo appartengono `ldap`, `meta`, `sql` e lo sperimentale `relay`; il primo serve a fare da proxy per un altro server, mentre il secondo può fare da proxy per un pool di server che condividono la gestione di una sezione di albero. Il terzo permette di utilizzare attraverso

[37] un database infatti deve comunque essere mantenuto in un qualche supporto.

[38] e le cosiddette funzionalità *ACID*, un acronimo che indica le quattro proprietà di *Atomicity*, *Consistency*, *Isolation*, e *Durability* che assicurano la consistenza dei dati e la possibilità di riportare il database in uno stato coerente anche in caso di terminazioni anormali.

[39] è ad esempio il default installato anche su Debian a partire da Lenny.

[40] cioè un accesso in scrittura deve avvenire senza che in quel momento venga compiuta sul database una qualunque altra operazione, sia di lettura che di scrittura.

Tipo	Significato
`bdb`	usa come *backend* Berkley DB 4.2 (vecchia interfaccia).
`hdb`	interfaccia alternativa (evoluzione di `bdb`) per Berkley DB .
`ldbm`	usa un qualunque database compatibile con l'interfaccia DBM.
`ldap`	fa da proxy verso un altro server LDAP.
`meta`	fa da proxy verso un gruppo di server LDAP.
`sql`	mappa i dati in un database relazionale su LDAP.
`relay`	fa da proxy nei confronti di un database locale (sperimentale).
`shell`	ottiene i dati da script di shell.
`perl`	ottiene i dati da script Perl.
`monitor`	riporta dati statistici del server.
`passwd`	riporta i dati di `/etc/passwd`.
`dnssrv`	riporta i dati da un DNS.
`null`	non fa nulla.

Tabella 1.9: I vari tipi di supporto per i dati di `slapd`.

LDAP i dati già presenti in un database relazionale,[41] Infine `relay` permette di creare un proxy al contenuto di un database locale (e può essere utilizzato per creare delle *viste* sullo stesso).

Al terzo gruppo appartengono `perl`, `shell`, `monitor`, `passwd`, `dnssrv` e `null`. I primi due consentono di abbinare ad ogni operazione l'esecuzione di script Perl o di shell, restituendo i relativi risultati; `monitor` è una interfaccia alle informazioni statistiche del server stesso, mentre `passwd` e `dnssrv` fanno da ponte per le informazioni mantenute nel file `/etc/passwd` e nei record `SRV` di un DNS. Infine `null` non fa proprio nulla.

Un server in cui di usano più supporti è poco comune, in quanto ha poco senso usare supporti diversi (si complica soltanto la gestione), a meno che l'uso non sia dettato da esigenze specifiche di compatibilità con vecchi dati o di accesso a formati particolari (ma converrebbe comunque effettuare una conversione). Può invece capitare di dover gestire più database (ad esempio per gestire indirizzari di diverse organizzazioni).

Il primo gruppo di direttive che esamineremo sono quelle globali, intese come quelle che sono applicabili a tutto il server. Si tenga presente comunque che alcune di esse, come le direttive per il controllo degli accessi (che tratteremo in sez. 1.3.1), normalmente non sono affatto utilizzate a livello globale quanto piuttosto a livello dei singoli database.[42]

Per semplificare la gestione dei file di configurazione inoltre si può usare la direttiva `include` che prende come argomento il nome di un file, il cui contenuto viene incluso automaticamente nel file principale come se fosse stato scritto direttamente all'interno di esso.[43] In generale questa direttiva viene utilizzata per includere nella configurazione i cosiddetti file di *schema* che contengono le definizioni degli oggetti che stanno nell'albero.[44]

Una direttiva molto importante (specie se si hanno elenchi molto grandi) è `sizelimit`, che impone un limite sul numero massimo di voci, da specificare come argomento, che il server restituisce come risultato di una ricerca; il default è di 500, e non è detto che sia sufficiente in

[41] come già accennato il modello di dati di un albero LDAP non si adatta alla struttura a tabelle di un database relazionale (una interessante discussione di questa problematica si trova nelle FAQ di OpenLDAP all'indirizzo `http://www.openldap.org/faq/data/cache/378.html`); pertanto questo *backend* non è un sostituto per mantenere i dati di un albero quanto una soluzione che consente, fornendo le opportune associazioni, di mappare il contenuto di un database relazionale su un albero.

[42] come è logico aspettarsi, dato che questi possono avere logiche di gestione diverse.

[43] la direttiva può essere usata ricorsivamente, ma attenzione, non esiste un limite sul numero di annidamenti e non c'è nessun meccanismo che rilevi la presenza di circoli viziosi in cui due file si includono a vicenda.

[44] questi si dichiarano a loro volta con delle direttive apposite, `objectclass` e `attributetype`, che hanno una sintassi specifica che non tratteremo, in genere infatti con OpenLDAP vengono distribuiti dei file di *schema* già pronti contenenti tutti gli oggetti principali necessari al normale funzionamento del server.

tutte le situazioni. Analoga è `timelimit` che impone un limite sulla durata del tempo speso dal server a fornire una risposta, questo fa riferimento al tempo reale e deve essere specificato in numero di secondi. Entrambe prevedono l'uso del valore `unlimited` per rimuovere i limiti. Le altre direttive principali sono riportate in tab. 1.10, un elenco completo è nella pagina di manuale accessibile con `man slapd.conf`.

Direttiva	Significato
`allow`	abilita una funzionalità specifica; valori possibili sono `bind_v2` che consente collegamenti con LDAPv2, `bind_anon_cred` che consente collegamenti non autenticati quando il DN è vuoto, `bind_anon_dn` che consente collegamenti non autenticati quando il DN non è vuoto, e `update_anon` che consente modifiche con collegamenti non autenticati.
`idletimeout`	specifica il numero di secondi da aspettare prima di chiudere forzatamente una connessione inattiva.
`include`	permette di includere le direttive da un altro file.
`loglevel`	abilita la registrazione di eventi ed operazioni sul sistema del *syslog* (di default viene usata la facility `LOCAL4`), usa come argomento un intero che specifica una maschera binaria i cui bit sono illustrati in tab. 1.5 (lo stesso valore usato con l'opzione `-d`), il valore di default è 128.
`sizelimit`	indica il numero massimo di voci che possono essere restituite in risposta ad una ricerca, il default è 500 (`unlimited` consente di restituire tutti i risultati).
`timelimit`	indica il limite massimo in secondi che il server può impiegare nel soddisfare una richiesta, il default è 30 secondi (`unlimited` indica di aspettare indefinitamente).
`argsfile`	indica un file in cui sono scritti gli argomenti a passati a riga di comando nell'avviare il server (quando non è eseguito interattivamente).
`disallow`	permette di disabilitare una delle funzionalità già elencate per `allow`.
`moduleload`	indica un modulo di estensione da caricare (in genere relativo al supporto per uno specifico tipo di database o per un *overlay*); prende o un pathname assoluto o uno relativo alla directory specificata con `modulepath`.
`modulepath`	indica la directory in cui vengono cercati i moduli di estensione.
`schemacheck`	richiede che il server, prima di accettare un oggetto, controlli che esso sia conforme agli *schema* utilizzati, deve essere mantenuto al valore di default che è `on`, onde evitare errori nell'immissione di nuovi oggetti, deprecata ed inutile nelle versioni più recenti di OpenLDAP.[45]
`pidfile`	indica il file in cui viene registrato il PID del server quando questo viene eseguito come demone in modalità non interattiva.

Tabella 1.10: Le direttive generali di `slapd.conf`.

Un esempio dell'uso di questo primo gruppo di direttive si trova nel seguente estratto del file di configurazione standard installato da una Debian Sid, da cui si sono rimossi per brevità righe vuote e commenti:

```
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema
pidfile         /var/run/slapd/slapd.pid
argsfile        /var/run/slapd.args
loglevel        0
modulepath      /usr/lib/ldap
moduleload      back_bdb
```

Si noti come nell'esempio prima si includono i vari file con gli schemi degli oggetti che immetteremo nel database, poi si impostano i file dove si troveranno memorizzate le informazioni

[45] la direttiva è ignorata nella versione 2.3 di OpenLDAP, ed è deprecata nella versione 2.4, dove il suo uso genera un errore.

relative al server, si disabilitano i messaggi di debug, si imposta la directory da cui caricare i moduli ed infine si attiva il supporto per Berkley DB.

Il secondo gruppo di direttive è quello delle direttive relative ad un determinato tipo di supporto, che vanno specificate dopo una direttiva `backend`. Si ricordi che in questo modo esse verranno applicate a tutti i database che usano quello stesso *backend*. Come per le direttive globali anche queste possono essere sovrascritte con valori diversi se riutilizzate in una sezione relativa ad uno specifico database che usi quel supporto, ed in effetti in genere è proprio in questo modo che vengono usate, ed è estremamente raro vederle applicate al di fuori di una sezione relativa ad un singolo database.

Dato che ogni *backend* ha delle caratteristiche proprie, ognuno definisce anche delle direttive specifiche, che saranno supportate soltanto se si è caricato il modulo che fornisce il supporto richiesto e per i database che lo utilizzano. Per ciascuno dei supporti descritti in tab. 1.9 esiste allora una apposita pagina di manuale `slapd-SUPPORTO`, con tutte le informazioni necessarie.

Dato che questo è il supporto più usato, noi tratteremo sommariamente solo le direttive relative al *backend* `bdb` (quelle di `hdb` sono le stesse); le direttive principali in questo caso sono: `cachesize` che indica quante voci mantenere in memoria e `directory` che indica in quale directory sono mantenuti i file del database. Le altre direttive principali sono riportate in tab. 1.11 mentre l'elenco completo è accessibile con `man slapd-bdb`.

Direttiva	Significato
`cachesize`	il numero di voci mantenute nella cache in memoria (se non specificata il valore di default è 2000).
`directory`	la directory in cui vengono mantenuti i file su cui sono registrati i dati (su Debian il default è `/var/lib/ldap`).
`mode`	i permessi con cui vengono creati i file dei dati, espressi in notazione ottale, come per `chmod` (il valore di default è `200`).
`index`	le opzioni di indicizzazione (vedi anche tab. 1.12).
`checkpoint`	la frequenza con cui viene scaricato su disco il registro delle transazioni, la direttiva prende due parametri, il primo che indica ogni quanti kilobyte ed il secondo che indica ogni quanti minuti deve essere eseguita l'operazione (il valore di default è nullo che significa che essi vengano ignorati).

Tabella 1.11: Le direttive per il supporto `bdb`.

In realtà le configurazioni più sofisticate relative a Berkley DB sono mantenute nel file `DB_CONFIG` nella directory dove i file del database perché le librerie di Berkley DB fanno riferimento ultimativo alle impostazioni che sono in questo file. Per questo il team di sviluppo di OpenLDAP non ha ritenuto opportuno inserire direttive specifiche dentro la configurazione del server,[46] e per la configurazione specialistica di `bdb` rimanda direttamente alla documentazione originale disponibile su `http://www.sleepycat.com/docs/`.

Un'altra direttiva definita a livello di supporto, ma disponibile per praticamene tutti i supporti relativi alla gestione dei dati, è `index`, che permette di definire quali informazioni indicizzare per un più veloce accesso ai dati; dato che comunque questa direttiva fa riferimento a dati specifici, anche questa non viene quasi mai utilizzata a livello di supporto generico, quanto piuttosto per ogni singolo database.

La direttiva prevede due argomenti, il primo è il nome dell'attributo da indicizzare mentre il secondo, opzionale, indica quale tipo di indicizzazioni effettuare, e prende una lista, separata da virgole, dei valori di tab. 1.12. Se come attributo si usa il valore riservato `default` si possono impostare una serie di indicizzazioni di default, che saranno applicate quando non si fornisce il valore di una indicizzazione specifica.

[46] questo avrebbe creato tutta una serie di possibili conflitti in caso di modifiche di questo file, difficili da rilevare.

Tipo	Significato
`pres`	indicizza per la presenza dell'attributo nel database qualunque valore esso abbia, permette di ottimizzare le ricerche in cui si richiedono tutti gli attributi di un certo tipo (ad esempio `cn=*`).
`eq`	indicizza per la ricerca di una eguaglianza esatta del valore dell'attributo (ad esempio `uid=piccardi`).
`sub`	indicizza per ricerca della presenza di una stringa all'interno del valore dell'attributo (ad esempio `cn=Simo*`). Questo tipo di indicizzazione può essere indicato in maniera specializzata usando al posto di `sub` i valori `subinitial`, `subfinal` e `subany` per specificare situazioni in cui la ricerca è eseguita su sotto-stringhe che stanno rispettivamente all'inizio (ad esempio `uid=sim*`), alla fine (ad esempio `uid=*ardi`) o nel mezzo (ad esempio `uid=*icc*`).
`approx`	indicizza per la ricerca di stringhe approssimativamente uguali, (cioè ricerche eseguite con la sintassi `cn~=simone`).
`none`	disabilita la generazione di indici (per attributi non indicizzabili come `jpegPhoto`).

Tabella 1.12: I vari tipi di indicizzazione usati come argomento della direttiva `index`.

La scelta dei giusti valori per le indicizzazioni è di fondamentale importanza per le prestazioni del server; pertanto è sempre opportuno indicizzare gli attributi sui quali si compiono le ricerche più frequenti;[47] non è però il caso di esagerare in quanto la creazione di indici inutili comporta spreco di memoria e rallenta le prestazioni in caso di scrittura, in maniera anche sensibile se il database contiene molti dati.[48] Inoltre è sempre opportuno selezionare il tipo di indici sulla base delle ricerche a cui vengono sottoposti i rispettivi attributi, ad esempio non ha senso indicizzare un attributo come `userPassword` per la ricerca di sotto-stringhe.

Il terzo gruppo di direttive sono quelle che eseguono le impostazioni specifiche di ciascun database. Esse sono utilizzabili soltanto all'interno di una sezione introdotta dalla direttiva `database` e sono indipendenti dal tipo di supporto utilizzato per lo stesso. Dato che nella maggior parte dei casi si ha un solo database che fa riferimento ad un unico supporto, in genere queste si trovano mescolate con le precedenti direttive specifiche del supporto nella stessa sezione.

Direttiva	Significato
`suffix`	imposta la radice associata al database, prende come argomento il *Distinguished Name* della stessa.
`rootdn`	imposta il *Distinguished Name* associato all'utente che amministra il database per il quale non valgono i controlli di accesso (vedi sez. 1.3.1).
`rootpw`	imposta la password dell'utente di amministrazione.
`readonly`	consente l'accesso al database in sola lettura.
`lastmod`	richiede che il server memorizzi nel database (in appositi attributi gestiti internamente) le informazioni relative all'ultima modifica di un oggetto, il valore di default è `on`.

Tabella 1.13: Le direttive generiche per i database.

Si ricordi infatti che un database corrisponde ad una specifica sezione di un albero LDAP, che viene definita dalla direttiva `suffix`. Questa definisce la radice per i dati presenti in quel database, e prende come parametro il relativo *Distinguished Name*.[49] Una direttiva `suffix` deve essere obbligatoriamente presente in ogni sezione `database`.

[47] vedremo con maggiore dettaglio in sez. 1.2.5 quando tratteremo i filtri di ricerca a quali indici queste corrispondono.

[48] come illustrato in sez. 1.1.1, le operazioni di scrittura su un database LDAP sono sporadiche: la gran del tempo impiegato da esse è proprio nella manutenzione degli indici che servono ad ottimizzare la lettura; evidentemente più dati sono mantenuti nel database, più costosa diventa l'operazione di aggiornamento degli indici.

[49] corrispondente al valore che si usa nella direttiva `BASE` per la configurazione del client.

Un elenco delle principali direttive generiche relative ad un database è riportato in tab. 1.13; la tabella contiene soltanto le direttive attinenti le funzionalità di base, tratteremo le direttive relative alle varie funzionalità avanzate più avanti, nelle relative sezioni. Un esempio dell'uso di alcune di queste direttive è il seguente proseguimento del precedente estratto dal file di configurazione standard installato da una Debian:

```
backend         bdb
database        bdb
suffix          "dc=truelite,dc=it"
directory       "/var/lib/ldap"
index           objectClass eq
lastmod         on
```

dove si dichiarano un supporto ed un database per il ramo di albero `dc=truelite,dc=it`; i file relativi saranno mantenuti in `/var/lib/ldap`, viene inoltre definito un indice di uguaglianza per gli attributi `objectClass` ed infine si richiede che il database registri i dati di ultima modifica degli oggetti.

1.2.4 Creazione e manutenzione di un elenco

Una volta installato e configurato il server per poter immettere le informazioni sull'elenco occorre predisporre l'infrastruttura che permetta di inserirle negli appropriati rami dell'albero. Nel caso di Debian installando il pacchetto `slapd` il sistema del `debconf` provvede a chiedere le informazioni necessarie alla creazione della struttura iniziale dell'albero, contenente soltanto la radice del dominio e l'utente usato per l'amministrazione. Negli altri casi occorre decidere quale sarà la radice dell'albero ed inserire le informazioni a mano.

In generale, anche se non è strettamente necessario,[50] è uso seguire una delle due strutturazioni illustrate in sez. 1.1.2. Per far questo occorre inserire nel database gli opportuni dati; questo si può fare in due modi: il primo prevede l'uso del protocollo stesso per inserire informazioni all'interno dell'albero, il che comporta l'uso di un qualunque client LDAP in grado di parlare con il server. Il secondo invece usa il programma di utilità `slapadd`, che fornisce un accesso diretto al database.

La differenza fra le due modalità è che nel primo caso si potrà operare anche da remoto, mentre nel secondo caso il comando potrà essere eseguito solo sulla macchina su cui si trova il database; inoltre, dato che `slapadd` opera sui dati in maniera indipendente, è sempre opportuno utilizzarlo quando il server non è attivo.[51] Il vantaggio è però che, con file di grosse dimensioni (ad esempio nel ripristinare un backup) il comando è molto più veloce e non necessita del collegamento al server con un utente dotato di sufficienti privilegi di accesso;[52] inoltre nel primo caso le informazioni di controllo su un oggetto come il tempo di ultima modifica[53] di solito non vengono impostate, in quanto la loro scrittura viene gestita autonomamente dal server; `slapadd`, operando direttamente sul database, può inserire facilmente anche queste informazioni.

Per inserire i dati con `slapadd` basta scrivere questi ultimi in formato LDIF. Il comando legge di default i dati dallo standard input, ma l'uso più comune è quello di usare l'opzione `-l` per farglieli leggere da un file. Si tenga presente che il comando si aspetta di ricevere i dati

[50] si potrebbe partire direttamente anche con un oggetto di tipo *Organization*, che è un contenitore sufficientemente generico, l'uso di una delle due strutturazioni di sez. 1.1.2 infatti ha senso solo se si ha intenzione di inserire l'elenco in un albero di visibilità globale, cosa che in realtà accade assai raramente.

[51] questo è sempre necessario quando si usa un supporto come `lbdm` che non supporta l'accesso concorrente in scrittura.

[52] il comando agisce direttamente sui file del database, l'accesso pertanto è diretto sul filesystem, e senza dover passare dall'autenticazione di LDAP; ovviamente occorre il permesso di scrittura sui file del database, che di norma ha soltanto l'amministratore.

[53] quello che il server mantiene automaticamente negli opportuni attributi interni quando si è impostata la direttiva `lastmod` a `on`.

nell'ordine in cui i database sono dichiarati in `slapd.conf`, che non esegue nessun controllo sulla presenza di una infrastruttura in cui inserire i dati,[54] né che questi siano consistenti con gli *schema* utilizzati; per questo motivo lo si usa principalmente per ripristinare i dati ottenuti da un dump eseguito da `slapcat`.

Qualora si voglia utilizzare un file di configurazione alternativo lo si può specificare con l'opzione `-f`; inoltre si può indicare direttamente su quale database operare, sulla base del valore della rispettiva radice, usando l'opzione `-b` (che prende come valore lo stesso parametro usato nella direttiva `suffix`). Le altre opzioni principali sono illustrate in tab. 1.14, per l'elenco completo si consulti la pagina di manuale del comando.

Opzione	Significato
`-l`	legge i dati in formato LDIF dal file specificato come parametro.
`-f`	usa come file di configurazione il file specificato come parametro.
`-b`	inserisce i dati nel database indicato dal suffisso usato come parametro.
`-d`	abilita la stampa di messaggi di debug delle operazioni, prende gli stessi valori illustrati in tab. 1.5.
`-c`	continua la scansione del file anche se riscontra degli errori (come voci già presenti che non saranno sovrascritte).

Tabella 1.14: Le opzioni del comando `slapadd`.

Come controparte di `slapadd` OpenLDAP fornisce anche il comando `slapcat`, che permette di estratte tutti i dati presenti in un database e stamparli a video in formato LDIF. Il comando è pertanto utilissimo per avere un backup completo del database pienamente riutilizzabile[55] con lo stesso `slapadd` per ricreare da zero un albero. Si tenga presente che mentre con `bdb` è possibile utilizzare questo programma anche a server attivo, con `ldbm` occorre prima fermare il server. Le principali opzioni di `slapcat` sono identiche a quelle di `slapadd`, riportate in tab. 1.14, solo che in questo caso con `-l` si indicherà il file su cui scrivere invece che quello da cui leggere; per la descrizione completa si faccia riferimento alla pagina di manuale.

Un ultimo comando di manutenzione che permette di operare direttamente sul database è `slapindex`, che viene usato per rigenerare gli indici del database rispetto ai contenuti correnti. In genere lo si usa tutte le volte che si vuole aggiungere (o rimuovere) un qualche indice. Se infatti ci si limita ad aggiungere o modificare una direttiva `index` in `slapd.conf` gli indici non vengono comunque ricostruiti fintanto che non lo si richiede esplicitamente con questo comando. Anche `slapindex` usa le stesse opzioni di `slapadd`, con lo stesso significato, tranne ovviamente l'opzione `-l`, che nel caso non ha senso.

In generale i comandi `slapadd`, `slapcat` e `slapindex` si usano per la manutenzione del database, in particolare `slapcat` è molto utile per creare dei backup del database da usare per ripristinare lo stesso in caso di errore o corruzione.[56] In tal caso basterà fermare il server, cancellare tutti i file usati per il supporto dei dati (nel caso di Debian il contenuto `/var/lib/ldap`), e poi ricreare il tutto con `slapadd`.

Qualora si voglia popolare il proprio albero inserendo delle singole voci si può sempre usare `slapadd`, ma come accennato questo presenta degli inconvenienti. Anzitutto con `slapadd` non si ha la generazione automatica delle informazioni di ultima modifica, che devono essere inserite a mano; inoltre non c'è controllo sulla corrispondenza dei dati agli *schema* utilizzati dal server,

[54] questo vuol dire che si avrà un errore se il dato esiste già o si cerca di inserirlo in una sezione dell'albero che ancora non esiste.

[55] per un backup basterebbe salvare i file del database (ed in particolare i file `is2entry`), ma questi saranno riutilizzabili solo con lo stesso tipo di supporto, con un dump in formato LDIF invece sarà sempre possibile ricreare il contenuto di un database su un altro server, con un supporto qualunque.

[56] ad esempio si può mettere uno script nei lavori giornalieri di *cron* che esegue `slapcat` per creare un file LDIF con il contenuto del proprio albero.

ed inoltre la sintassi dei file LDIF deve essere assolutamente corretta,[57] altrimenti nonostante l'apparente successo dell'operazione si rischia di non essere in grado di accedere alle informazioni; è pertanto il caso di usare `slapadd` soltanto con file LDIF di cui sia certa la validità (come quelli creati da `slapcat`).

Per questo motivo la modalità comune di inserimento di dati in un server LDAP è attraverso l'uso di un qualunque client che utilizzi le funzionalità di scrittura del protocollo stesso. Nel caso di Linux esistono parecchi programmi in grado di fare da client,[58] ma OpenLDAP fornisce anche un insieme di programmi di utilità a riga di comando per interagire con il server, attraverso le operazioni definite dal protocollo, di cui uno, `ldapmodify`, è quello che viene usato per manipolare le voci presenti su un server ed aggiungerne di nuove.

Per poter usare i comandi che operano sul server è però necessario autenticarsi sul server; l'autenticazione è un argomento complesso, che tratteremo in sez. 1.3.1, adesso basti sapere che esistono diverse modalità di autenticazione, per cui tutti i programmi client forniti da OpenLDAP supportano un gruppo di opzioni comuni relative alla autenticazione.

Nel nostro caso utilizzeremo solo quella che viene chiamata modalità di *autenticazione semplice* per attivare la quale occorre usare l'opzione `-x`,[59] mentre per specificare per conto di quale utente ci si vuole collegare si deve usare l'opzione `-D` seguita del DN che si vuole utilizzare. Se poi si usa l'opzione `-W` il programma chiederà l'immissione della password sul terminale, mentre usando `-w` questa potrà essere specificata direttamente sulla linea di comando.

Oltre a quelle relative all'autenticazione esistono anche altre opzioni comuni, ad esempio con `-H` si può specificare a quale server LDAP collegarsi usando la sintassi delle URI estese illustrata in sez. 1.2.2,[60] mentre con `-d` si può abilitare la stampa di messaggi di debug. Un elenco di tutte le principali opzioni comuni è riportato in tab. 1.15, al solito si rimanda alla lettura delle rispettive pagine di manuale per i dettagli.

Opzione	Significato
`-v`	stampa maggiori informazioni sullo standard output.
`-d`	abilita la modalità di debug, prende come parametro una maschera binaria secondo i valori illustrati in tab. 1.5.
`-x`	usa l'autenticazione semplice.
`-D`	specifica il DN da usare come utente per collegarsi al server.
`-W`	richiede l'immissione di una password per l'autenticazione semplice.
`-w`	specifica direttamente la password, passata come parametro, per l'autenticazione semplice.
`-y`	specifica un file contenente la password.
`-H`	specifica il server a cui collegarsi con la sintassi delle URL di LDAP.
`-h`	specifica il server a cui collegarsi per hostname o indirizzo IP (deprecato in favore di `-H`).
`-p`	specifica la porta a cui collegarsi sul server indicato con `-h` (deprecato in favore di `-H`).

Tabella 1.15: Le opzioni comuni dei programmi client di OpenLDAP.

Il comando `ldapmodify` è il comando generico che viene usato per modificare i contenuti di un albero LDAP; quando viene invocato con l'opzione `-a` o lanciato come `ldapadd` permette anche di aggiungere nuove voci. Ovviamente perché sia possibile effettuare le operazioni è necessario che l'utente con cui ci si collega al server (quello specificato con l'opzione `-D`) abbia i privilegi

[57] ad esempio la presenza di una riga vuota all'inizio di un file LDIF è causa comune di errori nella creazione del database.

[58] per gli amanti dell'interfaccia grafica `gq` è un ottimo client basato su GTK che fornisce una interfaccia grafica per l'accesso a qualunque tipo di informazione memorizzata su un server LDAP.

[59] la modalità impiegata di default è quella basata sulle librerie SASL, che non tratteremo.

[60] di default viene usato il server impostato in `ldap.conf`.

necessari. Si tenga presente che se non si specifica nessun utente viene fatto un collegamento anonimo che non consente di operare nessuna modifica.[61]

Il comando legge i dati dallo standard input, o da un file specificato con l'opzione `-f`. Oltre a quelle di tab. 1.15 il comando prevede l'opzione `-c` che permette di proseguire nelle modifiche anche se c'è stato un errore (il default è uscire al primo errore), e `-n` che si limita a mostrare quello che accadrebbe (permettendo così di rilevare eventuali errori) senza modificare effettivamente il contenuto del database. Al solito l'elenco completo delle opzioni è riportato nella pagina di manuale.

Fintanto che lo si usa per aggiungere una voce (con l'opzione `-a` o come `ldapadd`) i dati possono essere specificati in formato LDIF, in generale però il comando usa il formato del file di registro della replicazione (vedi sez. 1.4). Questo è un formato molto simile a LDIF, in cui gli oggetti ed i valori dei relativi attributi sono specificati nello stesso modo, ma prevede anche la presenza di una serie *pseudo-attributi* che indicano quali sono le operazioni da fare.[62] La descrizione del formato è disponibile nella pagina di manuale accessibile con `man slapd.replog`, non entreremo qui nei dettagli dato che l'uso più comune di `ldapmodify` è quello dell'aggiunta di nuove voci, per il quale basta usare LDIF.[63]

Si tenga presente però che `ldapmodify` non è in grado di modificare il DN di un oggetto; se cioè di un oggetto si vuole modificare l'attributo che mantiene il *Relative Distinguished Name* l'uso di `ldapmodify` darà luogo ad un errore.[64] Per compiere questa operazione occorre allora usare il comando apposito `ldapmodrdn`. Questo comando prende le stesse opzioni di `ldapmodify`, e prende come argomenti il DN dell'oggetto da modificare seguito da nuovo valore dell'RDN. Gli stessi valori si possono passare (ripetuti anche più volte, nel qual caso andranno separati da linee vuote) sullo standard input o su un file specificato con l'opzione `-f`.

Infine se si vogliono cancellare delle voci dall'albero OpenLDAP fornisce un comando specifico, `ldapdelete`. Questo prende come argomento uno o più *Distinguished Name* delle voci da cancellare. Se non si passa nessun argomento il comando cerca di leggere i suddetti DN dallo standard input. Oltre alle opzioni di tab. 1.15 anche `ldapdelete` prende le opzioni `-c` e `-n`, con lo stesso significato che hanno per di `ldapmodify`. Al solito l'elenco completo è nella pagina di manuale. Si tenga presente che anche in questo caso fintanto che non ci si collega al server usando un utente con privilegi appropriati le operazioni non saranno eseguite.

1.2.5 Interrogazioni e ricerche su LDAP

Per eseguire delle ricerche su un server, OpenLDAP fornisce il comando `ldapsearch`. A differenza dei precedenti comandi di scrittura in questo caso non è necessario collegarsi al server come utente, in quanto si esegue una semplice richiesta di lettura, che può essere consentita anche con un accesso anonimo.

Comunque qualora sia necessario eseguire un collegamento come utente si potrà fare ricorso alle opzioni già viste per i comandi di scrittura e riportate in tab. 1.15. Si tenga presente che in ogni caso si potranno ottenere soltanto quelle informazioni a cui i privilegi dell'utente con cui ci si collega danno accesso (e nel caso di accesso anonimo solo a quelle rese pubbliche).

[61] a meno di non aver effettuato una configurazione completamente sbagliata dei privilegi di accesso...

[62] in realtà alcuni attributi vengono ignorati, inoltre nel caso di `ldapadd` quello che accade è che se queste direttive speciali non sono presenti vengono utilizzati di default i valori che indicano l'aggiunta di una voce; dato che in questo modo l'informazione che resta è in formato LDIF, diventa possibile utilizzare direttamente un file in questo formato.

[63] se si devono modificare dei dati l'uso di `ldapmodify` è estremamente scomodo; data la presenza di innumerevoli alternative più semplici da usare, come `gq` per l'interfaccia grafica o `ldapvi` per la riga di comando, non vale la pena approfondire l'uso di un programma che ben difficilmente sarà usato a questo scopo.

[64] per farlo infatti occorre usare una operazione speciale, in quanto il DN è la chiave che consente a `ldapmodify` di accedere all'oggetto, ed è un parametro fisso della funzione che permette di modificare un oggetto.

Opzione	Significato
`-t`	scrive i risultati su un insieme di file temporanei (è utile per tutti quei casi in cui un attributo contiene dati non testuali).
`-A`	ricava solo la presenza degli attributi e non il loro valore.
`-L`	richiede che i risultati siano riportati in formato LDIF; specificato una sola restringe l'output a LDIFv1, usato due volte elimina i commenti, una terza volta non scrive la versione usata; il default è la versione estesa di LDIF.
`-S`	ordina i risultati sulla base dell'attributo passato come argomento.
`-b`	indica la base della ricerca da specificare con il corrispondente *Distinguished Name*.
`-s`	indica la profondità della ricerca, prende come argomento uno dei tre valori di tab. 1.17.
`-l`	indica un limite di tempo nell'esecuzione della ricerca (specificato in numero di secondi); un valore nullo soprassiede un eventuale limite specificato in `ldap.conf`.
`-z`	un limite sul numero di voci riportate come risultato della ricerca; un valore nullo soprassiede un eventuale limite specificato in `ldap.conf`.

Tabella 1.16: Le opzioni principali di `ldapsearch`.

Il comando prende come primo argomento un filtro di ricerca,[65] sulla cui sintassi torneremo più avanti, seguito da una lista opzionale di attributi. Se una o più voci corrispondenti al filtro di ricerca vengono trovate, il comando stampa sullo standard output i relativi risultati in formato LDIF. Se non si specifica un filtro verrà stampato il contenuto di tutto l'albero.[66]

Se si sono indicati degli attributi (per nome) verranno stampati solo i valori di questi ultimi, altrimenti verranno stampati tutti gli attributi presenti nell'oggetto.[67] Come attributo si può specificare il valore riservato "`+`" che identifica gli attributi speciali associati ad un oggetto,[68] che di default non vengono mai stampati; se si usa solo `+` verranno stampati solo gli attributi speciali, con "`+ -`" vengono stampati tutti gli attributi, speciali e non.

Con l'opzione `-b` inoltre si può restringere la ricerca ad una sezione dell'albero indicando la base della ricerca con il DN da cui partire (il default è usare la base dell'albero specificata in `ldap.conf`); si può poi definire la profondità a cui viene eseguita la ricerca con l'opzione `-s`, questa prende tre valori, illustrati in tab. 1.17. Le altre opzioni principali del comando sono riportate in tab. 1.16, al solito l'elenco completo è nella pagina di manuale.

Valore	Significato
`base`	indica il livello esatto del DN indicato.
`one`	indica il livello successivo a quello del DN indicato.
`sub`	indica tutti i livelli successivi a quello del DN indicato.

Tabella 1.17: Valori per il parametro che indica il livello di profondità di una ricerca.

I filtri di ricerca sono una delle funzionalità previste dal protocollo LDAP per ottimizzare le interrogazioni; il filtro infatti viene inviato direttamente al server come parte della richiesta, ed è quest'ultimo che si incarica di eseguire la ricerca con le condizioni in esso specificate, restituendo soltanto i risultati che corrispondono.[69] Per questo, oltre ad essere l'argomento principale di

[65] lo stesso che può comparire nelle URL estese viste in sez. 1.1.4.

[66] compatibilmente con il numero massimo di voci restituite dal server, secondo il valore della direttiva `sizelimit` vista in sez. 1.2.3.

[67] compatibilmente con i privilegi dell'utente con cui ci si è collegati, ad esempio nel caso di collegamento anonimo si potranno stampare solo o valori degli attributi dichiarati pubblici.

[68] quelli mantenuti direttamente dal server, come le date di modifica ecc. o l'attributo `hasSubordinates` che indica che un oggetto ha altri oggetti al disotto nella gerarchia dell'albero.

[69] questo consente di ridurre la quantità dei dati trasmessi, effettuando la selezione degli stessi direttamente sul server, uno degli errori più comuni per i programmatori alle prime armi con LDAP è proprio quello di farsi

`ldapsearch`, essi sono riconosciuti da qualunque client e vengono utilizzati tutte le volte che si vogliono definire dei criteri di ricerca per la selezione delle informazioni.

La sintassi dei filtri di ricerca è piuttosto complessa ed è stata standardizzata nell'RFC 2254; essi si basano su delle espressioni di base che poi possono essere combinate fra di loro; ogni espressione deve comunque essere racchiusa fra parentesi tonde, così come ogni combinazione di espressioni, pertanto un filtro sarà sempre espresso con una stringa del tipo: `(filtro)`.[70]

Le espressioni di base sono riportate in tab. 1.18; le prime quattro operano sui valori degli attributi, mentre la quinta seleziona tutti gli oggetti nell'albero che appartengono ad una certa `objectclass`.[71] Il tipo di ricerca che viene effettuata dipende dall'espressione utilizzata; inoltre le espressioni supportano la presenza del carattere jolly "`*`" che ne modifica il significato. Il filtro di default usato da `ldapsearch` quando non se ne specifica uno è `(objectclass=*)`, che indica tutti gli oggetti presenti nel database.

Espressione	Significato
`attr=val`	richiede l'uguaglianza del valore o la ricerca di una sotto-stringa se il valore contiene il carattere jolly "`*`" o la semplice presenza se si è usato solo "`*`".
`attr~=val`	esegue una ricerca di uguaglianza *approssimata* secondo uno degli algoritmi di somiglianza definiti dal protocollo.[72]
`attr>=val`	seleziona tutti i valori che siano lessicalmente uguali o *maggiori* di quello specificato.[73]
`attr<=val`	seleziona tutti i valori che siano lessicalmente uguali o *minori* di quello specificato.
`objectclass=val`	seleziona tutti gli oggetti della `objectclass` specificata.

Tabella 1.18: Espressioni di base di un filtro di ricerca.

Dato che l'espressione utilizzata in un filtro ha un diretto riscontro nel tipo di ricerca che il server si trova ad effettuare, e che per l'ottimizzazione delle ricerche sono necessari diversi metodi di indicizzazione del database (quelli indicati dai vari valori della direttiva `index` visti in tab. 1.12), è bene trattare in dettaglio tutti i possibili casi, in quanto la configurazione delle indicizzazioni da fare sul server è strettamente connessa al tipo di ricerche che vengono eseguite.

Se si richiede solo la corrispondenza di un attributo ad un certo valore, con una espressione del tipo "`(uid=piccardi)`", verrà eseguito un test di uguaglianza e si dovrà utilizzare una indicizzazione di tipo `eq`. Con l'uso del carattere jolly si possono però anche eseguire ricerche sulla presenza di una certa stringa, come `(mail=*@truelite.it)`, ed in questo caso si dovrà usare una indicizzazione di tipo `sub`.[74] Infine se si usa soltanto il carattere jolly, con una espressione del tipo di `(ou=*)` la ricerca sarà eseguita richiedendo solo la presenza di quell'attributo, e la indicizzazione dovrà essere di tipo `pres`; lo stesso avviene per tutte le ricerche sulle `objectclass`.

La potenza dei filtri di ricerca è che si possono combinare fra di loro più espressioni (di base, o costruite come combinazione di altre espressioni) utilizzando tre operatori logici che hanno rispettivamente il significato di AND, OR e NOT logico, secondo la sintassi illustrata in tab. 1.19. In questo modo è possibile ottenere dei filtri di ricerca anche molto complessi.

---

restituire tutti i dati per poi eseguire la ricerca sul lato del client.

[70] si tenga presente che, nel caso di uso di `ldapsearch` dalla riga di comando, le parentesi tonde sono caratteri riservati, per cui l'espressione del filtro dovrà essere adeguatamente protetta dall'interpretazione della shell.

[71] questo vale anche per le `objectclass` derivate da altre, ad esempio se si richiede `objectclass=person` saranno selezionati anche gli oggetti delle classi `inetOrgPerson` o `organizationalPerson` che dalla classe `person` sono derivate.

[65] in questo caso è necessaria la presenza di un indice di tipo `approx` (vedi tab. 1.12) per l'attributo in questione.

[73] questa regola di selezione richiede che l'attributo sia definito con una regola di ordinamento opportuna; in generale questo non avviene quasi mai e l'uso di un filtro che contenga una tale espressione, o la analoga `<=`, è estremamente raro.

[74] nel caso si usi BDB come *backend*, con le varie forme `subinitial`, `subfinal` e `subany` a seconda della posizione.

Operatore	Esempio	Significato
`&`	`(&(exp1)(exp2)(exp3))`	devono essere valide sia `exp1` che `exp2` che `exp3`.
`|`	`(|(exp1)(exp2)(exp3))`	devono essere valide o `exp1` o `exp2` o `exp3`.
`!`	`(!(exp))`	non deve essere valida `exp`.

Tabella 1.19: Operatori per le espressioni sui filtri di ricerca.

Ad esempio se si vogliono ricercare in un indirizzario tutte le persone il cui cognome inizia o per "S" o per "D", si potrà usare l'espressione:

```
ldapsearch -x "(|(sn=S*)(sn=D*))"
```

1.3 Configurazioni avanzate

Tratteremo in questa sezione la configurazione delle funzionalità più avanzate del server `slapd`, come le diverse modalità per l'autenticazione degli utenti, il controllo degli accessi ai dati mantenuti nel server, l'uso di TLS/SSL per la comunicazione fra client e server, i meccanismi per gestire la replicazione dei dati e la possibilità di eseguire riferimenti ad altri server.

1.3.1 Autenticazione e controllo degli accessi

In generale per potersi collegare ad un server LDAP è prima necessario autenticarsi in maniera opportuna, effettuando quell'operazione che viene chiamata *binding*, in cui ci si identifica come un *utente*, a cui saranno assegnati specifici privilegi di accesso, utilizzati poi per tutte le successive operazioni compiute sul server.

Il meccanismo in realtà è piuttosto complesso dato che OpenLDAP prevede due metodi di autenticazione. Il primo, richiesto anche dallo standard LDAPv3 ed usato di default da tutti i client a riga di comando distribuiti da OpenLDAP prevede l'uso di SASL,[75] che però necessita dell'installazione e della configurazione di apposite librerie,[76] ed una manutenzione separata delle informazioni riguardo gli utenti, che comporta quindi una amministrazione molto più complicata.

È supportato però anche un secondo metodo di autenticazione, chiamato *simple*, in cui un utente non è altro che un qualunque oggetto presente nel database che abbia un attributo di tipo `userPassword`. In questo caso l'utente che sarà identificato dal suo *Distinguished Name*, ed autenticato tramite la password mantenuta nel suddetto campo. Per semplicità tratteremo solo questo metodo, il cui svantaggio è che la password viene trasmessa in chiaro, e richiede pertanto una adeguata protezione del flusso dei dati (in sostanza l'uso di SSL).

Si tenga presente inoltre che il protocollo prevede anche la possibilità di un accesso anonimo non autenticato, che in genere viene usato per permettere a chiunque la lettura delle informazioni pubbliche mantenute sul server. Ovviamente un tale accesso dovrà avere privilegi estremamente ridotti ed essere limitato ad una sola sezione dell'albero.[77]

Infine, come illustrato in tab. 1.13, il server supporta per ciascun database la presenza di un utente speciale, definito dalla direttiva `rootdn` (che ne indica il *Distinguished Name*) per il quale non varranno i controlli di accesso che illustreremo in seguito e la cui password deve essere specificata direttamente nel file di configurazione tramite la direttiva `rootpw`.

Benché sia possibile impostare il valore della password direttamente tramite un file LDIF, la gestione risulterebbe piuttosto complessa a meno di non voler mantenere le password in chiaro

[75] acronimo di *Simple Authentication and Security Layer*, un metodo generico per aggiungere un meccanismo di autenticazione ai protocolli basati su connessione.

[76] nel caso specifico `libsasl`, fornita su Debian dal pacchetto `libsasl2`.

[77] non sarebbe molto saggio permettere a chiunque di scrivere sul proprio database, né di ottenere qualunque informazione in esso contenuta (in particolar modo le password degli utenti).

nel database;[78] OpenLDAP infatti supporta vari formati delle password, che prevedono differenti hash crittografici utilizzati per generarle.[79]

Per questo motivo è disponibile un apposito comando, `slappasswd`, che permette di generare il valore da inserire in un file LDIF generando l'opportuno hash crittografico della della password che viene immessa sul terminale. Il comando prende l'opzione `-s` per immettere direttamente la password come parametro e `-T` per leggerla da un file; altrimenti questa verrà chiesta sul terminale (due volte per conferma) senza che il suo valore sia visualizzato.

Parametro	Significato
`{SHA}`	usa l'algoritmo SHA-1.
`{SSHA}`	usa l'algoritmo Salted-SHA-1.
`{MD5}`	usa l'algoritmo MD5.
`{SMD5}`	usa l'algoritmo Salted-MD5.
`{CRYPT}`	usa la funzione `crypt`
`{CLEARTEXT}`	lascia la password in chiaro

Tabella 1.20: Parametri per l'opzione `-h` di `slappasswd`.

Infine l'opzione `-h` permette di scegliere l'algoritmo cui l'hash viene generato usando uno dei parametri elencati in tab. 1.20. Come risultato `slappasswd` stampa sullo standard output il valore da usare per `userPassword` all'interno di un file LDIF; ad esempio si avrà:

```
monk:/etc/ldap/schema# slappasswd
New password:
Re-enter new password:
{SSHA}xTelsDLvGxL2oqVrAS23HslitgVecAPs
```

Per evitare la noia di doversi generare a mano un file solo per modificare il valore di un attributo, fra i vari programmi client distribuiti da OpenLDAP c'è anche il programma `ldappasswd`, che permette di impostare o modificare la password associata ad un utente.

Oltre alle opzioni comuni di tutti i client di OpenLDAP già illustrate in tab. 1.15, il comando prevede l'opzione `-A` che richiede la immissione del precedente valore della password sul terminale, ma questa può essere specificata direttamente sulla riga di comando come parametro per l'opzione `-a`. Inoltre con `-S` si richiede l'immissione della nuova password su terminale, mentre la si può specificare direttamente come parametro per l'opzione `-s` o nel contenuto del file indicato con l'opzione `-t`.

Una volta che gli utenti sono presenti sul database la direttiva che consente di impostarne i privilegi di accesso sul server è `access`. La direttiva è una di quelle generali (cioè può essere utilizzata a livello di server) ma la si usa in genere, facendo riferimento agli utenti, a livello del singolo database. La sua forma generale, così come riportata nella pagina di manuale ad essa esplicitamente dedicata (accessibile con `man slapd.access`), è la seguente:

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

dove gli elementi fra `<>` indicano i diversi argomenti della direttiva che dovranno essere opportunamente specificati, le sezioni fra parentesi quadra sono opzionali, ed il simbolo `+` indica che esse possono essere ripetute più volte.

L'argomento `<what>` permette di specificare a quali oggetti si applica la direttiva di accesso e può assumere diverse forme. La forma più semplice è utilizzare il semplice "`*`" che indica tutte le voci presenti nel database. Una seconda forma è `attrs`, con cui si può specificare l'accesso ai singoli attributi, ad esempio con:

[78] pratica che è senz'altro sconsigliabile.

[79] questo in realtà va in contraddizione con lo standard stabilito nell'RFC 2256 che richiede che `userPassword` contenga testo in chiaro, una nuova specificazione nell'RFC 3112 prevede un nuovo attributo, `authPassword`, che però `slapd` non ha ancora implementato.

```
access to attrs=userPassword
```

si specificano i criteri di accesso per l'attributo che contiene la password, in qualunque oggetto esso si trovi. Con la stessa sintassi si possono specificare anche delle liste di attributi, da indicare con l'elenco dei rispettivi nomi separati da virgole. Allo stesso modo si può inserire nella lista il nome di una *objectClass*, usando il prefisso "`@`", nel qual caso la selezione si applicherà a tutti gli attributi in essa definiti; se invece si usa il prefisso "`!`" si selezioneranno tutti gli attributi che non sono nella *objectClass*.

Una forma alternativa dell'uso di `attrs` è quando questa viene usata per selezionare un solo attributo insieme alla specificazione `val` che consente di selezionare solo le voci in cui l'attributo specificato ha un determinato valore. Se ad esempio si volesse utilizzare l'attributo `organizationalRole` per dare accesso ad un ramo dell'albero soltanto a coloro che hanno un certo ruolo (identificato dal valore dello stesso), si potrebbe usare una direttiva del tipo:

```
access to dn.subtree="ou=ReservedContacts,dc=truelite,dc=it"
  by attrs=organizationalRole val="dirigente" write
  by * none
```

e si deve inoltre tenere presente che `val` può essere anche[80] usato nella forma `val.regex`, dove come valore si può usare una espressione regolare, che permette così di selezionare tutti gli attributi il cui valore corrisponda ad essa.

Nella lista degli attributi specificata da `attrs` si possono poi utilizzare anche due valori riservati, `entry`, che indica la voce stessa che contiene l'attributo, e `children` che indica tutte le voci sottostanti quella contenente l'attributo. Con queste due parole chiave è possibile impostare gli accessi selezionando le voci in base alla presenza di un attributo, senza dover ricorrere all'uso di un'espressione regolare che è molto più costosa in termini di prestazioni.

Una terza forma è quella in cui si usa `filter` per effettuare la selezione degli oggetti tramite un *filtro di ricerca*, utilizzando per questi ultimi la stessa sintassi usata con `ldapsearch` ed illustrata in sez. 1.2.5; un esempio potrebbe essere:

```
access to filter=(|(sn=S*)(sn=D*))
```

dove si selezionano tutti gli oggetti che hanno un attributo `sn` che inizia con la lettera S o con la lettera D.

La quarta ed ultima forma è quella che prevede la specificazione degli oggetti sulla base di espressioni indicanti il rispettivo *Distinguished Name*, questo è il meccanismo di selezione più complesso in quanto si possono usare diversi "*stili*" di selezione; la sua espressione generica è:

```
dn[.<dnstyle>]=<valore>
```

dove con `<dnstyle>` si indica uno degli stili di selezione elencati in tab. 1.21, mentre `<valore>` indica l'espressione di selezione. Questa per i primi 4 valori di tab. 1.21 assume il valore di un DN, nel caso di `regex` invece sarà una espressione regolare che sarà verificata contro tutti i DN degli oggetti dell'albero, selezionando quelli che corrispondono. Se non si specifica nessuno stile si applica `base`.

Quando si utilizza una espressione regolare si tenga presente che, dato un oggetto per cui essa corrisponde, corrisponderanno anche tutti gli oggetti che si trovano nella parte di albero sottostante il primo oggetto (in quanto il loro DN conterrà la parte già corrispondente), si ha cioè una selezione il cui *scope* è analogo a `subtree`.

Dato che l'uso di questa modalità comporta una scansione di tutti i DN degli oggetti, è il caso di usarla con parsimonia, se si vuole ad esempio prendere una sezione di albero al di sotto

[80] l'unica altra forma è `val.exact=valore`, che è quella di default, che pertanto non si è citata esplicitamente, in quando equivalente a `val=valore`.

Stile	Significato
`base`	indica una corrispondenza esatta con l'oggetto corrispondente al *Distinguished Name* specificato.
`exact`	sinonimo di `base`.
`one`	indica tutti gli oggetti contenuti nel livello dell'albero sottostante.
`onelevel`	sinonimo di `one`.
`subtree`	indica tutti gli oggetti contenuti nella sezione di albero che inizia dal *Distinguished Name* specificato.
`sub`	sinonimo di `subtree`.
`children`	indica tutti gli oggetti contenuti nella sezione di albero a partire dal livello successivo a quello del *Distinguished Name* specificato.
`regex`	indica tutti gli oggetti i cui *Distinguished Name* che corrispondono alla espressione regolare passata come argomento. È il valore di default, se non si specifica nulla viene assunta questa modalità.

Tabella 1.21: Stile di selezione degli oggetti per le direttive di accesso in base al *Distinguished Name*.

di un oggetto è il caso si usare `dn.subtree` che è molto meno pesante in termini di operazioni richieste al server.

L'argomento `<who>` permette di definire a chi assegnare l'accesso agli oggetti precedentemente selezionati, dato che il "*chi*" che identifica un utente può corrispondere ad oggetto mantenuto nell'albero stesso, di nuovo si potranno usare alcune modalità di selezione analoghe alle precedenti, come "`*`" che indica l'accesso generico per tutti e le varie forme basate su `dn` (che supportano gli stessi stili già illustrati in tab. 1.21) per effettuare una indicazione in base a dei DN.

Una delle caratteristiche più interessanti di questa selezione basata sul *Distinguished Name* è che se in una precedente selezione del `<what>` fatta con espressioni regolari si sono usati dei *subpattern*,[81] i valori di corrispondenza da essi ottenuti possono essere riutilizzati all'interno dell'espressione usata con `<who>` nella forma `$n`, dove `n` indica la n-sima corrispondenza nella selezione precedente. Diventa allora possibile definire criteri di accesso complessi come il seguente:

```
access to dn.regex="ou=Contacts,cn=([^,]+),ou=UserData,dc=truelite,dc=it"
  by dn.regex="uid=$1,ou=People,dc=truelite,dc=it" read
  by * none
```

in cui si dà accesso, a ciascun utente,[82] ad una distinta sezione dell'albero in cui si suppone siano stati inseriti i suoi contatti personali, evitando però che i singoli utenti possano accedere ai contatti personali degli altri.

L'argomento `<who>` supporta inoltre tre valori speciali, `self`, `anonymous` e `users`. In genere `self` è abbinato ad una selezione basata sugli attributi; con esso si indicano quali sono i permessi che un utente (cioè l'oggetto con il quale si è effettuato il collegamento al server) ha sui propri attributi.[83] Il caso classico infatti è:

```
access to attrs=userPassword
        by self write
```

[81] non è ovviamente possibile effettuare qui una trattazione approfondita delle espressioni regolari, argomento già di suo molto complesso, per un approfondimento si veda [RegExp].

[82] in questo caso si suppone che si disponga di una struttura dell'albero in cui i dati personali dei contatti di ciascun utente vengano mantenuti sotto il DN `ou=Contacts,cn=nomeutente,ou=UserData,dc=truelite,dc=it`; l'espressione regolare utilizzata seleziona il valore dell'attributo `cn` (che sarà il nome di un utente) composto da un numero arbitrario (ma maggiore di zero) di caratteri qualunque esclusa la virgola (che serve a separare i vari componenti del DN); il valore trovato è l'username che viene riutilizzato per identificare, nel ramo di albero in cui sono mantenuti gli utenti, per identificare quello a cui dare l'accesso alla rispettiva sezione di dati.

[83] cioè sugli attributi della voce corrispondente al proprio DN.

che consente ad un utente di modificare la propria password, senza però essere in grado di fare nulla con quella degli altri.

Con `anonymous` si indica l'utente generico non ancora autenticato; di norma si utilizza questo valore insieme al livello di autorizzazione `auth` per indicare un accesso in cui l'oggetto richiesto (usualmente `userPassword`) può essere acceduto solo a scopo di autenticazione. Se non si avesse questo tipo di accesso consentito per un utente anonimo diventerebbe impossibile effettuare un collegamento autenticato al server, non potendo disporre delle informazioni necessarie all'autenticazione.

Infine con `users` si indica il generico utente autenticato sul server, in questo caso si potrebbe dare l'accesso in scrittura ad una sezione particolare dell'albero (ad esempio quella dove sono mantenuti i contatti) ai soli utenti autenticati con una direttiva del tipo:

```
access to dn.subtree="ou=Contacts,dc=truelite,dc=it"
        by users write
```

La forma `dnattr` consente invece di utilizzare un attributo contenuto in una voce (quello specificato con `dnattr=attributo`) per dare l'accesso alla voce stessa. Questo è consentito se nell'attributo specificato è presente il valore del DN con cui viene effettuata la richiesta. Con questa forma si può specificare il *chi* può accedere direttamente all'interno della voce acceduta.

Un'altra forma molto utile è `group` che consente di dare accesso a tutti gli "*utenti*" di un "*gruppo*". In questo caso il gruppo viene costituito da una voce apposita voce che contenga le relative informazioni.[84] La forma generale di questa indicazione è:

```
group[/objectclass[/attributo]][.style]=pattern
```

dove `objectclass` indica la *objectclass* utilizzata per definire il gruppo e `attributo` l'attributo al suo interno che contiene i DN dei membri del gruppo. Un esempio di uso di questa forma è il seguente:

```
access to dn="ou=amministrazione,ou=Contacts,dc=truelite,dc=it"
  by group="cn=amministrazione,ou=Groups,dc=truelite,dc=it" read
```

Ulteriori forme speciali sono quelle che permettono di selezionare l'accesso in base ad informazioni esterne all'albero LDAP ottenute in genere direttamente dai parametri della connessione, come `domain`, `sockname`, `sockurl`, `peername` e `sockname`.

Tutte queste forme prevedono che si possano utilizzare stili diversi per indicare il valore del parametro passato come argomento, che come per `dn` devono essere specificati scrivendole come `forma.stile`; nel caso di `sockname`, `sockurl` gli stili possibili sono `exact` e `regex`, con `domain` a questi si aggiunge anche `sub` (o `subtree`) per indicare la scelta di tutto quello che sta sotto un dominio. Nel caso di `peername` si può avere invece lo stile `ip` in cui si specifica invece di un nome un indirizzo IP.

Come esempio di queste direttive, se si vuole fornire l'accesso anonimo in lettura ma solo per applicazioni residenti sul server, si potrà utilizzare una direttiva nella forma:

```
access to *
        by dn="cn=admin,dc=truelite,dc=it" write
        by peername.ip=127.0.0.1 read
        by self read
        by * none
```

con la quale per i collegamenti al server effettuati su `localhost` si garantisce comunque un accesso in lettura.

[84] il default è l'uso della *objectclass* `groupOfNames`, che nei suoi attributi `member` elenca i DN dei componenti del gruppo; si tenga presente che si possono usare altre *objectclass* con altri attributi, ma che i membri del gruppo devono essere specificati come DN; questo rende inutilizzabile a questo scopo l'uso di `posixGroup`.

Livello		Significato
`none`	–	nessun accesso.
`auth`	x	l'accesso è consentito solo internamente al server al solo scopo di autenticazione.
`compare`	c	consente l'accesso nelle ricerche ordinate.
`search`	s	consente l'accesso nelle ricerche.
`read`	r	consente la lettura dei dati.
`write`	w	consente la scrittura dei dati.

Tabella 1.22: Livelli di accesso usati dalla direttiva `access`.

Dopo aver specificato a chi si fa riferimento per i permessi d'accesso il successivo argomento `<level>` indica quale *livello di autorizzazione* si è assegnato. Questo argomento indica in sostanza quali sono le operazioni che è consentito eseguire sugli oggetti indicati nella direttiva. I vari livelli di autorizzazione sono riportati, in ordine crescente di priorità, in tab. 1.22. Se specificato semplicemente con uno dei valori della prima colonna della tabella, un livello di ordine superiore implica sempre anche quelli di ordine inferiore; se si vogliono specificare i singoli permessi separatamente lo si può dare usando una forma generica:

```
{=|+|-}{w|r|s|c|x}
```

con cui si possono assegnare, aggiungere o togliere i privilegi identificandoli con la rispettiva lettera nella seconda colonna di tab. 1.22.

Un esempio reale di utilizzo della direttiva `access` è riportato nel seguente estratto del file di configurazione standard installato da Debian, dal quale al solito si sono eliminati commenti e righe vuote:

```
access to attrs=userPassword
        by dn="cn=admin,dc=truelite,dc=it" write
        by anonymous auth
        by self write
        by * none
access to dn.base="" by * read
access to *
        by dn="cn=admin,dc=truelite,dc=it" write
        by * read
```

Infine si faccia attenzione a non inserire commenti a metà di una direttiva `access` come le precedenti, esse infatti sono suddivise su più righe sfruttando la caratteristica del file di configurazione che considera una prosecuzione della precedente una riga che inizia per spazio; se ad un certo punto ci si mette un commento la direttiva risulterà spezzata e gli effetti saranno tutt'altro che piacevoli.

Si noti come nell'esempio si sia usato per l'attributo `userPassword` un accesso di livello `auth` associato ad `anonymous` (la direttiva che indica un accesso non autenticato) per consentire l'autenticazione degli utenti,[85] e come con `self` si sia consentito agli utenti stessi di modificare la propria password.

L'esempio ci mostra una caratteristica fondamentale delle regole di accesso, che spesso provoca errori in quanto non se ne tiene dovutamente conto: le regole vengono esaminate nell'ordine in cui sono scritte, e quando si ha una corrispondenza sull'oggetto a cui si applicano (il `what`) la scansione si ferma immediatamente ed i permessi indicati vengono applicati.

Questo significa che occorre sempre mettere le regole più specifiche in testa, come viene fatto nell'esempio per la regola che riguarda solo l'attributo `userPassword`. Se prima di questa si fosse messa una regola generica come quella finale il risultato sarebbe stato che le successive

[85] questo perché prima di potersi collegare l'utente non è ancora autenticato, per cui l'accesso sarà in forma anonima, ma il server, per poter effettuare l'autenticazione, ha necessità di poter accedere alla password.

specificazioni non sarebbero mai state prese in considerazione e quindi chiunque avrebbe potuto leggere il dato della password e solo l'amministratore modificarlo.

Si noti anche come per le regole specifiche sia necessario ripetere l'indicazione degli stessi criteri di accesso (come quello per l'amministratore) elencati nella successiva regola generica, questo perché, se non ci fossero, essendo ignorate in caso di corrispondenza le regole seguenti, ci si sarebbe trovati con un utente amministrativo incapace di accedere autonomamente in scrittura al campo `userPassword` di tutti gli utenti.

1.3.2 La configurazione per l'uso di SSL/TLS

La configurazione di OpenLDAP per l'uso di TLS è tutto sommato abbastanza semplice, fintanto che si dispone già di un certificato valido. Il problema è che in generale non è così ed è invece necessario predisporre tutto il necessario per la gestione di una *Certification Authority* (in breve CA) che firmi i certificati delle varie macchine, specie se oltre al server presso i client si vogliono anche autenticare i client presso il server.[86]

Per far questo si possono usare gli strumenti forniti con le librerie OpenSSL; il pacchetto Debian relativo è `openssl` che contiene l'omonimo programma, più un apposito script in Perl, sotto `/usr/lib/ssl/misc/CA.pl`,[87] che permette di semplificare la creazione della *Certification Authority*, facendo da interfaccia ad `openssl` per la gestione delle varie operazioni da eseguire per creare la propria personale CA.

Alternativamente si può usare uno qualunque dei programmi che si appoggiano ad `openssl` per la gestione di una *Certification Authority*, per gli amanti dell'interfaccia grafica vale la pena citare `tinyca` (installabile in Debian dall'omonimo pacchetto), che usa le librerie GTK; nonostante alcuni difetti dell'interfaccia, questo programma è uno dei più completi per quanto riguarda la gestione di una CA. Non essendo scontata la disponibilità di una interfaccia grafica, in seguito faremo riferimento soltanto allo script `CA.pl` distribuito insieme ad `openssl`.[88]

In generale è buona norma creare i certificati e le chiavi su una macchina dedicata, non connessa alla rete e tenuta in luogo sicuro. L'accesso alle chiavi della CA permette infatti di compromettere tutte le comunicazioni. Non disponendo di una macchina dedicata si può eseguire l'operazione su un portatile isolato dalla rete, e salvare tutti i file su un dischetto; i paranoici possono anche stampare per ulteriore precauzione certificati e chiavi (per riporli in luogo sufficientemente sicuro), e poi cancellare il tutto.[89]

Lo script `CA.pl` genera i certificati usando alcuni valori predefiniti che sono impostati nelle variabili definite all'inizio dello stesso. In particolare può essere utile modificare la durata dei certificati che creeremo. Questo si realizza modificando il valore della variabile `$DAYS`, che viene utilizzata per specificare di quanti giorni deve essere la validità del certificato. Un possibile esempio è quello in cui si è modificato la relativa riga in:

```
$DAYS="-days 1095";
```

che permette di ottenere un periodo di durata dei certificati di tre anni (1095 giorni) al posto dell'anno (365 giorni) che viene utilizzato nella installazione di default di `openssl`.

Una volta operate le modifiche ritenute necessarie alle variabili dello script, il primo passo della procedura è quello di creare l'infrastruttura che conterrà i vari dati (fra cui certificati e chiavi) della nostra nuova Certification Authority personale. Questo si fa con l'opzione `-newca`; un esempio di output del comando è:

[86] quest'ultimo caso comunque è piuttosto raro, per i problemi che vedremo più avanti.

[87] se non si dispone del Perl, esiste uno shell script, `CA.sh`, che esegue le stesse operazioni.

[88] le istruzioni, a parte il nome del comando, sono le stesse qualora si volesse utilizzare `CA.sh`.

[89] meglio se lo si fa usando un programma di cancellazione sicura come `wipe` (su Debian installabile dall'omonimo pacchetto), per maggior sicurezza.

```
parker:~/CertAuth# /usr/lib/ssl/misc/CA.pl -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Using configuration from /usr/lib/ssl/openssl.cnf
Generating a 1024 bit RSA private key
..++++++
........++++++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Italia
Locality Name (eg, city) []:Firenze
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Truelite s.r.l.
Organizational Unit Name (eg, section) []:CA
Common Name (eg, YOUR name) []:monk.truelite.it
Email Address []:piccardi@truelite.it
```

premendo invio si genera un certificato con il nome standard. Lo script richiede poi una password per la chiave segreta della CA, che va confermata. Infine si devono immettere le informazioni previste dallo standard X.509, che saranno riportate nel certificato della CA, procedendo come mostrato nell'uscita a video del comando riportata qui sopra.

Tutti i file verranno creati in una sottodirectory `./demoCA` a partire da quella in cui si esegue lo script: la chiave segreta verrà creata in `./demoCA/private/cakey.pem` ed il certificato con la chiave pubblica in `./demoCA/cacert.pem`; si possono poi controllare i dati relativi a quest'ultimo con il comando:

```
parker:~/CertAuth# openssl x509 -text -noout < demoCA/cacert.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 0 (0x0)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=IT, ST=Italia, L=Firenze, O=Truelite SRL, OU=Internal CA, \
CN=truelite.it/emailAddress=info@truelite.it
        Validity
            Not Before: May 18 20:54:43 2004 GMT
            Not After : May 16 20:54:43 2014 GMT
        Subject: C=IT, ST=Italia, L=Firenze, O=Truelite SRL, OU=Internal CA, \
CN=truelite.it/emailAddress=info@truelite.it
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
...
```

Il passo seguente è quello di creare i certificati per ciascuna macchina che dovrà essere autenticata. Se si vuole semplicemente assicurarsi una trasmissione criptata questo non è necessario e basta creare un solo certificato per il server, lo diventa qualora si intenda autenticare il server presso i client e viceversa.

Il primo passo per creare un nuovo certificato, è quello di generare un certificato di richiesta; questo si fa usando lo script con l'opzione `-newreq`, l'output del comando è:

```
parker:~/CertAuth# /usr/lib/ssl/misc/CA.pl -newreq
Using configuration from /usr/lib/ssl/openssl.cnf
Generating a 1024 bit RSA private key
....++++++
...........++++++
writing new private key to 'newreq.pem'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Italia
Locality Name (eg, city) []:Firenze
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Truelite s.r.l.
Organizational Unit Name (eg, section) []:IT
Common Name (eg, YOUR name) []:parker.truelite.it
Email Address []:piccardi@truelite.it

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Request (and private key) is in newreq.pem
```

Lo script prima chiede la password da usare per proteggere per la chiave privata del certificato, che va confermata, poi passa alla richiesta dei dati dello standard X.509 da inserire nello stesso, analogamente a quanto fatto per il certificato della CA.

In questo caso occorre essere precisi nella definizione del Common Name, esso infatti deve corrispondere esattamente al FQDN (*Fully Qualified Domain Name*, il nome completo dell'estensione di domino) della macchina su cui si installerà il certificato, altrimenti la connessione con SSL/TLS non funzionerà.[90]

Alla fine delle operazioni il certificato di richiesta e la chiave privata ad esso associata saranno memorizzati nel file `newreq.pem`. Si può controllare il certificato di richiesta con il comando:

```
parker:~/CertAuth# openssl req -text -noout < newreq.pem
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=IT, ST=Italia, L=Firenze, O=Truelite SRL, OU=VPN, \
CN=fw.truelite.it/emailAddress=info@truelite.it
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
...
```

Una volta creato il certificato di richiesta, questo deve essere firmato dalla Certification Authority, in modo da ottenere un certificato valido. Questo si fa con l'opzione `-sign`, l'output del comando è:

[90] OpenLDAP è estremamente pignolo riguardo a questo, il client deve essere in grado di risolvere l'*hostname* del server e questo deve corrispondere esattamente al nome riportato nel certificato, pena il fallimento della connessione.

```
parker:~/CertAuth# /usr/lib/ssl/misc/CA.pl -sign
Using configuration from /usr/lib/ssl/openssl.cnf
Enter PEM pass phrase:
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'IT'
stateOrProvinceName   :PRINTABLE:'Italia'
localityName          :PRINTABLE:'Firenze'
organizationName      :PRINTABLE:'Truelite s.r.l.'
organizationalUnitName:PRINTABLE:'IT'
commonName            :PRINTABLE:'parker.truelite.it'
emailAddress          :IA5STRING:'piccardi@truelite.it'
Certificate is to be certified until Aug 21 07:42:59 2004 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Signed certificate is in newcert.pem
```

il comando chiederà la password della CA per firmare il certificato, e chiederà conferma riguardo la firma e la memorizzazione del nuovo certificato (lo script mantiene tutti i file necessari alla CA per tenere traccia dei vari certificati firmati, che vanno in `./demoCA/newcerts`). Il certificato firmato sarà salvato nel file `newcert.pem`. Anche questo può essere controllato con il comando:

```
parker:~/CertAuth# openssl x509 -text -noout < newcert.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=IT, ST=Italia, L=Firenze, O=Truelite SRL, OU=Internal CA, \
CN=truelite.it/emailAddress=info@truelite.it
        Validity
            Not Before: May 18 20:55:51 2004 GMT
            Not After : May 18 20:55:51 2005 GMT
        Subject: C=IT, ST=Italia, L=Firenze, O=Truelite SRL, OU=VPN, \
CN=fw.truelite.it/emailAddress=info@truelite.it
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
...
```

Perché `slapd` possa usare il certificato occorre però anche la sua chiave privata, e, non potendo essere effettuata una richiesta di password nel lancio di un demone, questa deve essere fornita in chiaro, pertanto occorrerà convertire la chiave contenuta in `newreq.pem`, che è cifrata, con il comando:

```
parker:~/CertAuth# openssl rsa < newreq.pem > newkey.pem
read RSA key
Enter PEM pass phrase:
writing RSA key
```

dando la password del certificato. Si tenga presente che il file `newkey.pem` contiene la chiave privata in chiaro, e pertanto deve essere protetto scrupolosamente, cambiandone immediatamente in permessi in `400` (sola lettura per il proprietario) ed assegnato a root.[91]

[91] questo andrebbe fatto contestualmente all'esecuzione del comando, utilizzando `umask` per rimuovere dai permessi di default quelli di lettura.

Qualora si voglia accedere al servizio con diversi nomi a dominio la cosa si complica, perché non è possibile usare più certificati per uno stesso server, e le connessioni SSL vengono identificate sulla base del nome a dominio[92] contenuto nel relativo certificato, che deve corrispondere esattamente. In questo caso occorre allora fare uso di una estensione prevista dallo standard X.509v3, che permette di specificare all'interno di uno stesso certificato dei nomi a dominio alternativi. Questo però comporta alcune modifiche alla procedura appena illustrata.

Per tutto ciò occorre impostare `openssl` perché quando si firmano i certificati in essi venga inserito il campo che definisce ai nomi alternativi. Questa procedura però non si può automatizzare con `CA.pl`, per cui per ciascun certificato occorre modificare il file di configurazione di `openssl`, (su Debian è `/etc/ssl/openssl.cnf`) dove viene definito il valore di questo campo. Per questo, prima di firmare il certificato di richiesta,[93] occorre inserire nella sezione `[ usr_cert ]` di `openssl.cnf` una riga analoga alla seguente:

```
subjectAltName = DNS:ldap.truelite.it,DNS:parker.truelite.it
```

che definisce il campo `subjectAltName` (nel default installato da Debian è commentato) inserendovi la lista di nomi a dominio alternativi con i quali si vuole contattare il server. La lista deve essere specificata con il formato illustrato nell'esempio,[94] come elenco, separato da virgole, dei vari nomi a dominio preceduti dall'indicatore "`DNS:`", è anche possibile indicare tutto un sottodominio utilizzando il metacarattere "`*`" per creare quello che viene denominato un certificato *wildcard*.[95]

In questo modo, una volta firmato il certificato, il nome alternativo sarà inserito nel file `newcert.pem`, dove potremo verificare la presenza di un campo aggiuntivo del tipo:

```
...
            X509v3 Subject Alternative Name:
                DNS:ldap.truelite.it
...
```

Una volta creati i certificati si può passare alla configurazione di `slapd`; il primo passo è copiarli sulla macchina che ospiterà il server; in genere si possono mantenere insieme agli altri file della configurazione di SSL/TLS, nella directory deputata ai certificati, `/etc/ssl/certs`, nel caso li si sono rinominati opportunamente come:

```
[root@havnor certs]# ls -l
total 16
-r--r--r--    1 root     root         1326 Aug 13 16:16 cacert.pem
-r--r--r--    1 root     root         3711 Aug 13 16:16 ldapcert.pem
-r--------    1 root     root          887 Aug 13 16:16 ldapkey.pem
-r--------    1 root     root         1720 Aug 13 16:16 ldapreq.pem
```

Dopo di che si tratta di configurare il server per fargli usare detti certificati; per questo `slapd.conf` prevede un gruppo di direttive specifiche per la configurazione di SSL/TLS, che consentono ad esempio di indicare al server dove si trovano i precedenti file. Si sono illustrate le direttive più rilevanti in tab. 1.23, al solito l'elenco completo è riportato nella pagina di manuale di `slapd.conf`, nella sezione *TLS options*.

Un estratto della sezione di `slapd.conf` in cui si sono inserite le direttive necessarie per abilitare l'uso di TLS/SSL, in modo da utilizzare i precedenti certificati, è allora il seguente:

[92] per essere precisi dal FQDN.

[93] lo si può fare anche prima di creare la richiesta, la cosa è ininfluente al fine di avere il valore nel certificato finale.

[94] oltre a quello riportato, che specifica un nome a dominio, si possono usare anche altri indicatori, come `URL:`, `email:` o `IP:` usati rispettivamente per pagine web, indirizzi di posta elettronica e indirizzi IP; questi vari tipi di indicatori possono anche essere mescolati fra loro.

[95] si tenga presente che in questo caso è necessario usare questo tipo di valori come `subjectAltName` e non direttamente come nomi per il *Common Name* principale, dato che l'uso dei certificati *wildcard* è supportato in maniera ufficiale sono in questo caso (ad esempio è solo così che funziona con GnuTLS).

Direttiva	Significato
`TLSCertificateFile`	file in cui è contenuto il certificato che identifica il server.
`TLSCACertificateFile`	file in cui è contenuto il certificato della CA che certifica il server.
`TLSCertificateKeyFile`	file che contiene la chiave del certificato del server.
`TLSVerifyClient`	richiede che anche il client si identifichi con SSL presso il server.

Tabella 1.23: Le principali direttive di `slapd.conf` per l'uso di TLS/SSL.

```
# For SSL/TLS authentication
TLSCertificateFile      /etc/ssl/certs/ldapcert.pem
TLSCertificateKeyFile   /etc/ssl/certs/ldapkey.pem
TLSCACertificateFile    /etc/ssl/certs/cacert.pem
```

Si tenga presente che per poter utilizzare TLS occorre anche avviare opportunamente il server (si ricordi quanto detto in sez. 1.2.1); nel caso di Debian per questo si può usare la variabile di ambiente `SLAPD_SERVER` definita in `/etc/default/slapd` che permette di indicare esplicitamente le modalità di avvio del server, una configurazione tipica potrebbe essere la seguente:

```
SLAPD_SERVICES="ldap://127.0.0.1/ ldaps:///"
```

con cui si mette il server in ascolto in chiaro per le connessioni locali e si usa TLS/SSL per tutte quelle remote.

Oltre al server, per poter utilizzare TLS, occorre anche fornire le adeguate impostazioni al client, si dovranno cioè utilizzare le opportune direttive di configurazione previste per `/etc/ldap/ldap.conf`. In questo caso, nello specificare quale è il server a cui si fa riferimento, sarà preferibile usare la direttiva `uri`, che permette la sintassi estesa delle URL di tab. 1.1; nel nostro caso dovremo specificare l'indirizzo del server come `ldaps://ldap.truelite.it` eliminando le direttive `host` e `port`.

Inoltre il file di configurazione deve anche indicare il certificato della *Certification Authority* che certifica l'autenticità del server, e una direttiva che impone il riconoscimento del server. Per questo occorrerà aggiungere all'esempio di file di configurazione visto in sez. 1.2.2 una sezione apposita con un contenuto del tipo:

```
TLS_CHECKPEER  yes
TLS_CACERT     /etc/ssl/certs/cacert.pem
```

Un elenco delle principali direttive usate per la configurazione dell'accesso via SSL ad un server LDAP da parte di un client è riportato in tab. 1.24;[96] per l'elenco completo al solito si faccia riferimento alla pagina di manuale di `ldap.conf`.

Direttiva	Significato
`TLS_CERT`	file in cui è contenuto il certificato che identifica il client.
`TLS_CACERT`	file in cui è contenuto il certificato della CA che certifica il server.
`TLS_KEY`	file che contiene la chiave del certificato del client.
`TLS_REQCERT`	indica il livello di controllo che si richiede sui certificati del server.
`TLS_CHECKPEER`	richiede la identificazione del server, può assumere i valori `yes` e `no`.

Tabella 1.24: Direttive per la configurazione di `ldap.conf` per l'uso di TLS/SSL.

Se si è attivata la richiesta di verifica del client da parte di `slapd`[97] occorre configurare il client perché invii il suo certificato. Per ciascun client si dovranno pertanto ripetere le precedenti operazioni per creare un certificato firmato da una CA riconosciuta del server.[98] Anche qui deve avere cura di usare come *Common Name* il nome a dominio (in forma di FQDN) del client.

[96] le direttive sono riportate in maiuscolo, ma si possono specificare anche in caratteri minuscoli.
[97] lo si fa con la direttiva `TLSVerifyClient` illustrata in tab. 1.23.
[98] in genere si usa sempre la stessa, ma non è necessario.

In questo caso si ha il problema che a livello di `ldap.conf` è possibile specificare solo un certificato per ciascuna macchina, che sarà mantenuto in una opportuna directory (in genere su usa comunque `/etc/ssl/certs/`). Questo significa però che la chiave privata del certificato, per poter essere utilizzata, deve essere leggibile da ogni utente, e non è una buona pratica di sicurezza il fatto che un singolo utente possa avere pieno accesso ad un certificato che identifica una intera macchina (ed influire quindi anche sulle capacità degli altri utenti).

Per questo motivo una configurazione in cui si richiede il riconoscimento dei client non dovrebbe essere fatta sul file di configurazione generale `ldap.conf`, ma essere specificata per il singolo utente; le librerie di OpenLDAP prevedono infatti anche un file di configurazione personale che è `.ldaprc`,[99] in cui l'utente può specificare le sue opzioni; in questo caso dovrebbero esservi inserite delle direttive del tipo:

```
TLS_CACERT      cacert.pem
TLS_CERT        my_cert.pem
TLS_KEY         my_key.pem
TLS_REQCERT     hard
```

che permettono l'uso di un certificato personale (la prima è opzionale se il file della CA è già stato impostato in `ldap.conf`). Questo significa però che se si deve autenticare il client rispetto al server, occorrerà anche dotare ciascun utente di un suo certificato, rendendo ovviamente la gestione degli utenti molto più macchinosa.

Una volta che si sia configurato il server e le librerie per l'uso di SSL se ne potrà verificare il funzionamento direttamente con `ldapsearch`, si può richiedere l'uso di SSL direttamente sulla riga di comando con l'opzione `-H` usando la apposita URL estesa (vedi tab. 1.1), un esempio potrebbe essere:

```
ldapsearch -x -H ldaps://ldap.truelite.it
```

In caso di problemi (quello più comune è la non corrispondenza fra il nome a dominio con cui si contatta il server e quello presente nel certificato) si può usare il precedente comando con l'opzione `-d 1` per aumentare l'informazione di debug e verificare le motivazioni di eventuali malfunzionamenti.

1.3.3 Gli *overlay*

Con la versione 2.2 di OpenLDAP è stato introdotto un ulteriore meccanismo di estensione e modularizzazione delle funzionalità del server nella forma dei cosiddetti *overlay*, che vanno a costituire una estensione del modello che consente di usare diversi *backend* per la memorizzazione dei dati e consentono di ottenere un potente mezzo per aumentarne le funzionalità.

In sostanza il meccanismo prevede la possibilità di interporsi fra il funzionamento del *frontend* del server LDAP, che riceve le richieste, le decodifica e le passa al *backend* e quest'ultimo che fornisce relative le risposte. Con questa interposizione sia le richieste che le risposte possono essere intercettate in modo da potervi eseguire sopra le operazioni volute. In questo modo si può personalizzare il funzionamento di un *backend* senza doverne creare una versione modificata, utilizzare le estensioni solo dove necessario.

Inizialmente questa funzionalità era utilizzabile solo in corrispondenza ad uno specifico *backend*; partire dalla versione 2.3 è divenuto possibile utilizzare gli *overlay* in maniera generica, ad esempio per ottenere funzionalità indipendenti dai diversi *backend* o anche utilizzare le informazioni della richiesta per selezionare l'uso di uno specifico *backend* piuttosto di un altro.

[99] in realtà il file di configurazione può essere uno qualunque fra `$HOME/.ldaprc`, `$HOME/ldaprc` e `$HOME/$LDAPRC`, e si può usare la variabile di ambiente `LDAPRC` anche per specificare un file a piacere.

Gli *overlay*, come i *backend*, sono realizzati in forma di moduli,[100] e prima di poterli utilizzare devono essere caricati con l'uso della appropriata direttiva `moduleload`. Una volta che il modulo sia stato caricato le funzionalità devono essere attivate per ciascun *backend* o *database* attraverso la direttiva `overlay`. Questa è una direttiva che opera a livello di database, e dovrà essere specificata per ciascun database per cui si vuole utilizzare il relativo *overlay*. Pertanto se si vuole utilizzare l'*overlay* `unique` si dovranno specificare le direttive:

```
moduleload unique
overlay unique
```

La direttiva richiede come argomenti il nome dell'*overlay* che si vuole usare, che corrisponde al nome del relativo modulo caricato in precedenza. In tab. 1.25 si è riportata una lista dei principali *overlay* disponibili, con una breve descrizione degli stessi, l'elenco completo può essere ottenuto consultando la relativa pagina di manuale, accessibile con `man slapd.overlays`; la documentazione dei singoli overlay è invece mantenuta in altrettante pagina di manuale, accessibili a loro volta con `man slapo-nomeoverlay`.

Nome	Descrizione
`syncprov`	Implementa il *provider* per una replicazione con `syncrepl` (vedi sez. 1.4.3).
`ppolicy`	Implementa politiche di controllo di qualità sulle password degli utenti LDAP.
`valsort`	Consente di richiedere uno specifico ordinamento dei valori di un attributo a valori multipli in risposta alle ricerche.
`unique`	Consente di garantire l'unicità del valore di un attributo in un ramo di albero (ad esempio per richiedere che un attributo che identifica un utente sia unico).
`constraint`	Consente di impostare dei vincoli sui valori che possono essere assegnati ad un attributo (ad esempio per forzare l'inserimento di valori corretti per codici postali, numeri telefonici, nomi a dominio, ecc.).
`accesslog`	Consente di registrare tutti gli accessi ad un dato database su un altro database, in modo che sia possibile ottenerli da quest'ultimo attraverso richieste LDAP; in questo modo si possono revisionare col protocollo stesso le operazioni fatte su un albero (il meccanismo viene usato per la cosiddetta *delta-replication*, vedi sez. 1.4.4).

Tabella 1.25: Principali *overlay* disponibili per `slapd` (usabili come argomenti per la direttiva `overlay`).

Una delle caratteristiche più interessanti degli *overlay* e che se ne possono utilizzare anche più di uno ripetendo la direttiva per ciascuno di essi; in questo caso che gli *overlay* vengono *impilati* uno sull'altro nell'ordine in cui li si dichiarano, quindi verrà eseguito per primo quello dichiarato per ultimo. Il concetto è quello che ogni richiesta viene passata allo strato degli *overlay* prima di arrivare al database, il primo la riceve, la tratta e passa il risultato al successivo e via seguendo; le risposte fanno il percorso inverso. Questo consente di combinare fra loro le diverse funzionalità fornite dai singoli *overlay*.

1.3.4 Ottimizzare le prestazioni

Da fare.

1.4 La replicazione

Affronteremo in questa sezione l'argomento della replicazione, che è una delle caratteristiche più interessanti di LDAP. La replicazione è il meccanismo che consente di distribuire su più server

[100] in sostanza si tratta di librerie condivise, installate nella directory dei moduli (usualmente `/usr/lib/ldap`) insieme a quelle dei *backend*.

mantenendole coerenti fra loro le informazioni di un albero di dati LDAP. Esamineremo le varie modalità con cui la replicazione viene effettuata e le relative configurazioni.

1.4.1 Il meccanismo della replicazione

Una delle caratteristiche più interessanti di LDAP è la facilità con cui, grazie al supporto dei *referral* (vedi sez. 1.1.4) fornito dal protocollo stesso, è possibile creare e mantenere aggiornate automaticamente varie copie del database, così da poter distribuire le richieste o creare dei backup automaticamente. Questo viene realizzato attraverso una funzionalità che viene chiamata in modo generico "*replicazione*", che consente di creare dei server secondari (i cosiddetti *slave*) che mantengono automaticamente le stesse informazioni presenti su un server primario (il cosiddetto *master*).

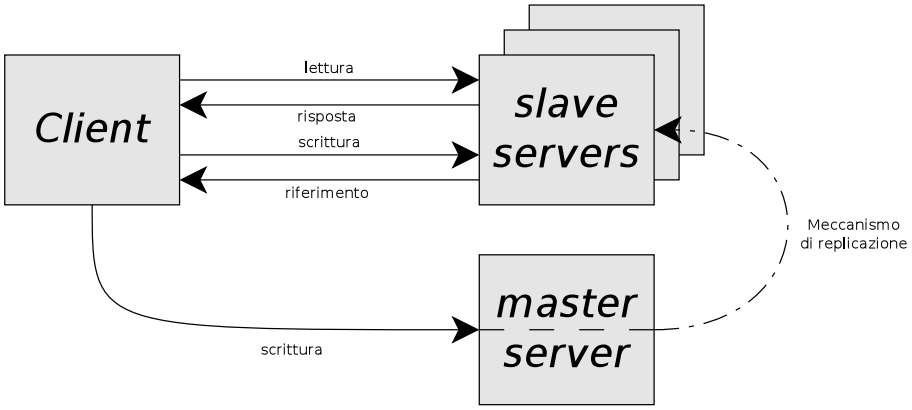


Figura 1.5: Il funzionamento della replicazione.

Il funzionamento generale della replicazione, sommariamente illustrato in fig. 1.5, prevede che quando un client effettua una richiesta per leggere dei dati di un albero tramite un secondario quest'ultimo risponda direttamente, quando invece si vogliono effettuare delle modifiche il secondario risponderà al programma che ha richiesto la modifica inviando un riferimento al server primario,[101] in modo che la richiesta possa essere fatta direttamente a lui. Se si è attivato un opportuno meccanismo di replicazione, una volta che il server primario ha effettuato la modifica, sarà compito di tale meccanismo far sì che i dati modificati vengano aggiornati sui vari secondari, attraverso i canali di comunicazione usati per la replicazione.

Una volta completato il ciclo di aggiornamento tutti i secondari avranno i dati coerenti fra loro e con il primario; si tenga presente però, come accennato in sez. 1.1.1, che il protocollo LDAP non esclude delle inconsistenze temporanee nei dati, e questo vale anche per differenze fra il primario ed i vari secondari, e anche di questi ultimi fra loro, dovute appunto al fatto che il meccanismo appena illustrato non dà nessuna garanzia sulla immediatezza della sincronizzazione dei dati.

Come per le modalità con cui vengono mantenuti i dati dell'albero, anche per il meccanismo della replicazione il protocollo LDAP non prevede una specifica modalità realizzativa, infatti tutto quello che serve è che i vari server gestiscano in maniera coerente i riferimenti e mantengano coordinate le informazioni. Esistono pertanto diverse implementazioni della replicazione, che tratteremo nelle prossime sezioni.

Infine, a partire dalla versione 2.4 di OpenLDAP viene supportata anche una diversa modalità di replicazione in cui non si ha più la restrizione di un singolo *master* che raccoglie e distribuisce tutte le modifiche agli *slave* come illustrato in fig. 1.5. La nuova modalità di replicazione,

[101] il meccanismo è sempre quello dei *referral* che abbiamo illustrato in fig. 1.4, in questo caso però i riferimenti vengono inviati solo, a cura del secondario, per le richieste di scrittura.

denominata *multi-master*, consente infatti ad ogni server di agire sia come *master*, eseguendo direttamente le richieste di modifica per poi trasmetterle agli altri server con gli opportuni meccanismi di sincronizzazione, sia come *slave* per ottenere le modifiche effettuate allo stesso modo dagli altri server.

1.4.2 La replicazione con `slurpd`

Fino alla versione 2.3 di OpenLDAP, quando è stato introdotto il nuovo meccanismo denominato *syncrepl*, la replicazione veniva realizzata esclusivamente attraverso l'utilizzo sul server principale di un demone dedicato, `slurpd`, che si incaricava di inviare ai secondari le modifiche effettuate sulla copia principale dell'albero, secondo lo schema illustrato in fig. 1.6. A partire dalla versione 2.4 l'uso di `slurpd` non è più supportato ed occorre utilizzare obbligatoriamente *syncrepl*, che tratteremo in sez. 1.4.3.

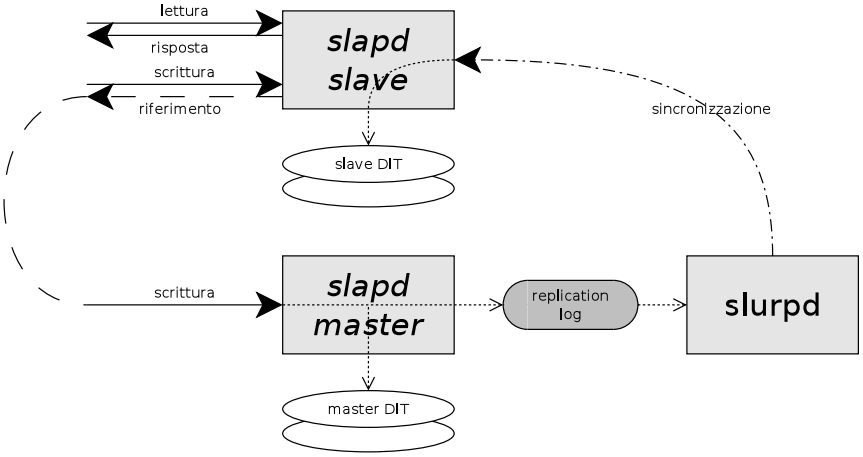


Figura 1.6: Il funzionamento della replicazione con `slurpd`.

Il funzionamento della replicazione con `slurpd` prevede che il demone rilevi le modifiche ai dati che vengono effettuate sul server principale leggendole su un apposito file, detto *giornale di replicazione*. Questo file viene opportunamente creato e mantenuto dal server `slapd` del *master*, che vi salva tutte le modifiche effettuate al database con la relativa marca temporale. Quando `slurpd` rileva nel *giornale di replicazione* delle modifiche non ancora applicate (usando la marca temporale) il programma crea un file di lock e le trascrive in una copia privata, e poi le invia ai secondari.[102] Nel caso di Debian viene usato il file `slurpd.replog`, mantenuto, come gli altri file usati da `slurpd`, sotto `/var/spool/slurpd/replica/`.

L'opzione `-r` di `slurpd` permette di indicare manualmente un *giornale di replicazione* alternativo a quello di default (letto dalla configurazione), ed in genere la si usa insieme all'opzione `-o` che attiva la cosiddetta modalità *one-shot*, in cui il programma legge il file, applica le modifiche ed esce subito dopo. Questa è la modalità che si usa per eseguire manualmente (a demone fermo) eventuali modifiche rimaste bloccate per un qualche errore. Infine l'opzione `-d` attiva la modalità di debug; in questo caso il programma resta agganciato al terminale[103] e verranno stampate a video tutte le informazioni di controllo specificate dalla maschera binaria passata come parametro dell'opzione.[104]

Come demone `slurpd` non ha un suo file di configurazione; per gestire la replicazione normalmente viene usato lo stesso `slapd.conf`, in cui devono essere presenti le apposite direttive di

[102] semplicemente effettuando delle richieste di scrittura LDAP, che in questo caso, essendo i secondari configurati per riconoscerle ed accettarle, verranno eseguite.

[103] a meno di non aver usato `-o`, nel qual caso termina comunque dopo aver eseguito le modifiche.

[104] per i valori di questo parametro si consulti la pagina di manuale, accessibile con `man slurpd`.

replicazione riassunte in tab. 1.26.[105] Con l'opzione `-f` comunque si può indicare al programma l'uso un file di configurazione alternativo.

Direttiva	Significato
`replica`	Definisce i parametri che indicano un server secondario da usare come replica.
`replicationinterval`	Indica l'intervallo in secondi usato da `slurpd` nel controllare la presenza di cambiamenti sul giornale di replicazione.
`replogfile`	Indica il file sul quale viene mantenuto il giornale di replicazione.
`updatedn`	Indica sul secondario quale DN è utilizzato per ricevere i dati di replicazione.
`updateref`	Indica sul secondario quale a quale server primario fare riferimento in caso di richieste di modifica.

Tabella 1.26: Le principali direttive usate per la replicazione con `slurpd`.

Il primo passo da fare per impostare una replicazione è di configurare il primario perché generi le informazioni ad essa necessarie, per questo c'è la direttiva `replogfile`, che permette di indicare un file sul quale sarà scritto il cosiddetto *giornale di replicazione* (o *replication logfile*). La direttiva richiede come parametro il nome del file che verrà scritto da `slapd`, e poi letto da `slurpd`.

La seconda direttiva necessaria sul primario è `replica`, che permette di specificare tutti i dati che indicano a `slurpd` verso chi inviare le informazioni di replicazione. La direttiva prevede la indicazione di una serie di argomenti nella forma `nome=valore`, separati da spazi. Di questi l'unico obbligatorio è `uri` che serve ad indicare la URI del server secondario su cui verranno replicati i dati. Se si hanno più secondari basterà usare una diversa direttiva `replica` per ciascuno di essi.

Normalmente oltre alla URI del server deve essere anche specificato, con l'argomento `binddn`, l'utente con cui collegarsi sul secondario per scrivere i dati,[106] con `bindmethod` il metodo di autenticazione, e con `credentials` il valore della password da usare. Un elenco degli altri principali argomenti della direttiva `replica` è illustrato in tab. 1.27.

Argomento	Significato
`uri`	indica la URI del server secondario, con la stessa sintassi illustrata in tab. 1.1.
`suffix`	indica quali sezioni dell'albero devono essere replicate, se non specificato è l'intero albero, altrimenti si possono indicare più sezioni attraverso il rispettivo DN.
`binddn`	indica (come DN) l'utente con il quale effettuare il collegamento al server secondario; ci si accerti che questo abbia privilegi sufficienti per poter scrivere le relative informazioni.
`bindmethod`	indica la modalità con cui effettuare il collegamento al server, può assumere i valori `simple` o `sasl`.
`credentials`	indica la password dell'utente con cui effettuare il collegamento.
`attr`	indica gli attributi da replicare.

Tabella 1.27: Principali argomenti della direttiva `replica`.

Un possibile esempio delle direttive di configurazione da inserire su un server primario per la creazione di una replica di un albero è allora quello contenuto nel seguente estratto di `slapd.conf`:

[105] si tenga presente che queste direttive riguardano comunque anche il funzionamento del demone principale `slapd`.

[106] che in genere dovrebbe essere diverso da quello di amministrazione del database, dato che un secondario può ospitare anche altri alberi; qualora il secondario sia semplicemente usato come replica del primario si può anche usare lo stesso utente.

```
replogfile /var/lib/ldap/replog

replica uri=ldaps://slave.truelite.it:636
        binddn="cn=replicatore,dc=truelite,dc=it"
        bindmethod=simple credentials=password_segretissima
```

Nel caso di Debian una volta aggiunte dette istruzioni lo script di avvio del servizio resta `/etc/init.d/slapd`, che quando rileva la presenza di una direttiva `replica` nel file di configurazione si prende cura di lanciare nell'ordine corretto sia `slapd` che `slurpd`.[107]

La configurazione dei secondari deve ovviamente indicare al demone `slapd` che gira su di essi il loro stato di *slave*, in modo che le richieste di modifica dei dati che dovessero pervenire per la parte di albero replicata vengano reinviate al primario, così che la propagazione delle stesse possa avvenire sempre in maniera unidirezionale dal primario al secondario. Questo viene eseguito con la direttiva `updateref` che prende come argomento la URL del primario che il secondario deve restituire alle richieste di scrittura.

Inoltre occorre segnalare a `slapd` il suo stato di secondario e quale utente è usato per ricevere i relativi dati; questo si fa tramite la direttiva `updatedn` che deve corrispondere esattamente all'utente specificato nella direttiva `replica` usata nel primario. Un esempio delle direttive da inserire nel file `slapd.conf` del secondario, corrispondenti alle precedenti viste per il primario, è allora il seguente:

```
updatedn   cn=replicatore,dc=truelite,dc=it
updateref  uri=ldaps://master.truelite.it:636
```

e si ricordi che l'utente qui specificato deve avere privilegi sufficienti a poter scrivere i dati.

Si tenga presente infine che se come negli esempi `slurpd` necessita di collegarsi via SSL ai vari secondari per la trasmissione delle modifiche, occorrerà anche aver definito opportunamente in `ldap.conf` le direttive illustrate in sez. 1.3.2 per indicare il certificato della Certification Authority. Si tenga inoltre presente che qualora si siano fatti degli errori o sia caduta la rete, per cui `slurpd` non riesce a contattare il server secondario, le modifiche fatte al primario restano comunque memorizzate nel giornale di replicazione, e verranno applicate automaticamente una volta ristabilita la connessione.

La caratteristica della replicazione effettuata con `slurpd` è che quest'ultimo trasmette solo le modifiche effettuate sul server primario, perché il meccanismo funzioni occorre allora che prima dell'avvio della replicazione il secondario sia sincronizzato col primario. Se si è configurato il sistema per eseguire la replicazione fin dall'inizio e si è creato il database da zero non ci sono problemi, altrimenti si avranno degli errori tutte le volte che `slurpd` si trova ad eseguire operazioni facenti riferimento a sezioni di albero, dati o attributi inesistenti sul secondario. In tal caso quello che accade è che `slurpd` genera dei file `.rej` contenenti le modifiche che sono state *rigettate*,[108] così che sia possibile effettuare un inserimento manuale una volta corrette le inconsistenze.

Quello della sincronizzazione iniziale è il maggior limite del meccanismo illustrato, infatti il caso più comune è quello in cui si vogliono replicare i dati di un server già presente. Questo vuol dire che prima di poter utilizzare il meccanismo occorrerà caricare sul secondario gli stessi dati del primario. L'operazione di per sé è relativamente semplice, in quanto basta fermare il primario, eseguire un dump dei dati con `slapcat` e caricare quest'ultimo sul secondario con `slapadd`. Dove possono sorgere problemi è quando si è eseguita una replica parziale, e ci si trova con delle inconsistenze di dati, pertanto anche la precedente operazione deve essere fatta con il primario fermo, in modo da essere sicuri che il secondario abbia gli stessi dati del primario.

Uno degli errori più comuni è quello in cui si vuole aggiungere un nuovo ramo dell'albero, e si crea detto ramo *prima* di aver impostato la sua replicazione. In tal caso quello che accade

[107] che significa lanciare prima `slapd` e poi `slurpd` all'avvio e bloccare prima `slurpd` e poi `slapd` allo stop.
[108] sempre sotto `/var/spool/slurpd/replica/`, almeno per Debian.

è che le successive immissioni dei dati falliranno, in quanto sul secondario non esiste la radice del nuovo ramo. Inoltre per il funzionamento è cruciale il corretto ordinamento delle operazioni ed il fatto che non si perda per troppo tempo la connessione fra i server, altrimenti le differenze possono crescere oltre il limite in cui `slurpd` è in grado di gestirle.

Inoltre tutte le volte che per un qualche motivo la sincronizzazione fallisce, le differenze devono essere riapplicate manualmente con la procedura descritta; in casi come questo diventa molto utile conoscere il formato dei dati del giornale di replicazione, e del file `replog.status` che viene utilizzato da `slurpd` per indicare lo stato delle operazioni di replicazione. Questo è sostanzialmente, a parte alcune estensioni, lo stesso formato adottato per i file usati con il comando `ldapadd` per eseguire modifiche ad un albero LDAP, ed è descritto in una specifica pagina di manuale, accessibile con `man slapd.replog`

1.4.3 La replicazione con *syncrepl*

Date le molteplici problematiche che presenta l'uso di `slurpd` a partire dalla versione 2.3 di OpenLDAP è stato introdotto un nuovo meccanismo per gestire la replicazione denominato *syncrepl*. Il meccanismo è basato sull'utilizzo di uno specifico protocollo di sincronizzazione,[109] denominato *LDAP Content Synchronization Protocol* e standardizzato nell'RFC 4533, che consente di estendere opportunamente le richieste di ricerche per farsi fornire i cambiamenti apportati o presenti su un ramo di albero, così da poter mantenere la propria copia in uno stato coerente, in maniera indipendente dalle differenze correnti al momento della sincronizzazione.

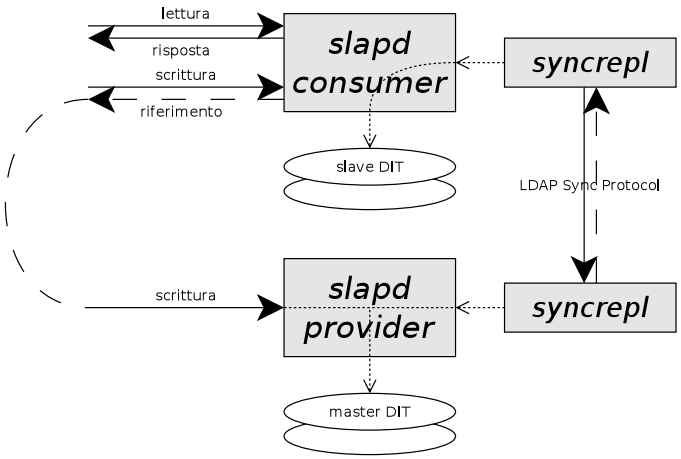


Figura 1.7: Il funzionamento della replicazione con *syncrepl*.

Questo protocollo consente in sostanza di effettuare delle ricerche speciali che trasmettano soltanto le differenze presenti fra due versioni dello stesso albero, fornendo così una grande flessibilità nelle modalità con cui si può effettuare la replicazione,[110] compresa la possibilità di alternative al modello classico di un server *master* che pilota il contenuto di vari server *slave*, con la possibilità di avere sistemi con più server *master* o server in cui i ruoli di *master* o *slave* vengono mescolati. Per questo motivo nella documentazione di OpenLDAP è stata introdotta una terminologia alternativa, in cui si indicano come *provider* i server che forniscono le informazioni di replicazione, mentre si chiamano *consumer* quelli che le ricevono, ed un server può agire sia da *consumer* verso un server, che da *provider* per un altro.

[109] il meccanismo è comunque realizzato in forma generica e può essere esteso ad altri protocolli qualora questi venissero ideati.

[110] si tratta comunque di ricerche LDAP, che supportano quindi l'indicazione di filtri, profondità, attributi da restituire, ecc. come illustrato in sez. 1.1.4.

Il protocollo prevede che sia sempre il *consumer* a collegarsi al *provider* eseguendo poi la sincronizzazione secondo lo schema illustrato in fig. 1.7.[111] Il protocollo prevede che il *consumer* possa fornire nelle richieste un opportuno *cookie* di sincronizzazione che indica lo stato del suo database (in sostanza quando è stata fatta l'ultima sincronizzazione), in modo che il *producer* possa fornire tutte le modifiche effettuate in seguito. Se non viene inviato nessun *cookie* il primario invierà tutto il contenuto del database.[112] É compito del primario mantenere le informazioni di sincronizzazione ed aggiornare ad ogni modifica il cosiddetto *Change Sequence Number* (o CSN) dell'albero, una sorta di marca temporale che consente al primario di identificare quali sono le informazioni da inviare al secondario nella fase di sincronizzazione.[113] Alla fine di ciascuna fase di sincronizzazione viene inviato al secondario anche il CSN corrente, così che in seguito questo non debba ripartire da zero.

Nel suo uso più elementare il meccanismo di *syncrepl* fornisce due diverse modalità di eseguire la sincronizzazione. La prima modalità, denominata *refreshOnly*, prevede connessioni periodiche da parte del *consumer* verso il *provider* per ottenere le differenze intercorse. La seconda modalità, denominata *refreshAndPersist*, prevede che il *consumer* esegua una sincronizzazione iniziale ma poi resti connesso e che poi sia il *provider* ad inviare le ulteriori modifiche quando queste vengono effettuate, utilizzando la connessione rimasta attiva.

In entrambi i casi non è più necessaria la presenza di un demone esterno e tutte le operazioni vengono eseguite direttamente da `slapd`, che deve essere opportunamente configurato. Quanto necessario per utilizzare la replicazione sul *provider* viene realizzato con un apposito *overlay*, `syncprov`, il cui compito è fornire il supporto per l'uso del protocollo *LDAP Content Synchronization Protocol*. Questo *overlay* può essere usato in combinazione con un qualunque database che supporti l'uso degli attributi speciali `entryUUID` e `entryCSN`,[114] che sono quelli consentono rispettivamente di identificare in maniera univoca ogni voce e di tener conto di quando è stata modificata.

Inoltre siccome tutte le ricerche utilizzate nel protocollo di sincronizzazione fanno ampio uso dei dati memorizzati nei due attributi citati, per avere buone prestazioni è sempre opportuno eseguire una indicizzazione di questi attributi aggiungendo alle direttive di indicizzazione del database su cui si vuole utilizzare *syncrepl* una riga del tipo:

```
index entryCSN,entryUUID eq
```

L'*overlay* `syncprov` mantiene inoltre un'ulteriore attributo speciale, `contextCSN`, aggiunto alla radice del database, che indica lo stato attuale dell'albero e consente di tracciare le differenze introdotte. Per evitare operazioni aggiuntive su disco,[115] questo attributo viene letto all'avvio di `slapd` e mantenuto in memoria. Di default il valore viene aggiornato solo alla chiusura del server, ma l'uso della direttiva `syncprov-checkpoint` consente di richiedere salvataggi periodici dello stato; la sua sintassi, come quella delle altre principali direttive fornite dall'*overlay* `syncprov` sono riassunte in tab. 1.28.

Riassumendo, per configurare `slapd` per fare da *provider* ed agire come *master* per la replicazione occorrerà anzitutto richiedere il caricamento del modulo per l'*overlay* `syncprov` nella

[111] questo ha un preciso significato dal punto di vista dei protocolli di rete, è cioè il *consumer* che deve iniziare la connessione verso il *producer*, ed un eventuale firewall interposto fra i due deve consentire questa comunicazione.

[112] cosa che può portare ad un carico pesante in caso di database molto grandi; in tal caso è opportuno effettuare un dump e ripristinarlo come per `slurpd`, il vantaggio è che lo si può fare senza spegnere il primario, perché eventuali modifiche successive verranno sincronizzate automaticamente.

[113] esistono in realtà due diverse modalità di sincronizzazione, usate alternativamente o insieme a seconda dei casi: la cosiddetta *delete phase* in cui si trasmettono DN ed `entryUUID` delle voci cancellate e la *present phase* in cui si trasmettono integralmente i dati modificati e solo DN ed `entryUUID` di quelli presenti (da cui si desume quali sono quelli cancellati).

[114] il loro uso deve anche essere abilitato con l'uso della direttiva `lastmod on`, ma in genere non è necessaria una configurazione esplicita dato che questo è il valore di default.

[115] in particolare il blocco sul database di *backend* necessario per un aggiornamento affidabile.

Direttiva	Significato
`syncprov-checkpoint`	configura i salvataggi periodici delle informazioni di modifica del database; richiede due argomenti numerici indicanti rispettivamente ogni quante operazioni ed ogni quanti minuti operare il salvataggio (se non specificata non viene effettuato nessun salvataggio periodico).
`syncprov-sessionlog`	indica la lunghezza del registro delle operazioni di scrittura da mantenere in memoria; richiede come argomento il numero di operazioni.

Tabella 1.28: Principali direttive utilizzabili con l'*overlay* `syncprov`.

sezione generale del file di configurazione, e poi abilitarne l'uso e configurarlo per i vari database di cui si vuole effettuare la replicazione; in sostanza la configurazione dovrà contenere delle direttive analoghe alle seguenti:

```
...
moduleload syncprov
...
...
# provider DB
database hdb
suffix          "dc=truelite,dc=it"
lastmod on
...
index           entryCSN,entryUUID eq
overlay syncprov
syncprov-checkpoint  100  10
syncprov-sessionlog  100
...
```

Per la configurazione di un server secondario come *consumer* invece non serve nessun *overlay* ma la funzionalità viene gestita direttamente da `slapd` tramite la direttiva `syncrepl`; da dichiarare per ogni database viene che si vuole gestire come replica. La direttiva è simile alla analoga `replica` usata con `slurpd`, e prevede l'uso di una serie di argomenti da indicare nella stessa forma `nome=valore`, alcuni dei quali sono identici ed hanno lo stesso significato di quelli usati per `replica`.

In tab. 1.29 si sono riportati gli argomenti principali, ma la direttiva è complessa, in quanto consente di eseguire la replicazione in modalità diverse con configurazioni molto sofisticate (su cui torneremo in sez. 1.4.4) per cui vedremo qui soltanto quelli relativi ad una configurazione di base in cui la si utilizza per mantenere sincronizzato un albero secondario in maniera analoga a come si fa con `slurpd`.

Qualunque sia la modalità di replicazione che si vuole attivare, è comunque necessario fornire a `syncrepl` alcuni argomenti; il primo di questo è `rid`, un identificatore di tre cifre decimali che deve essere associato a ciascuna dichiarazione della direttiva (si può scegliere un valore qualsiasi purché univoco) in modo da distinguere, qualora come possibile se ne usi più di una, le diverse istanze della replicazione.

L'argomento `provider` deve invece indicare, nella forma di una URI di LDAP (la stessa usata per la direttiva `uri` di `ldap.conf`) quale è il server che deve essere usato come *provider*. Se necessario (in genere lo è sempre) si dovranno poi specificare con `binddn`, `bindmethod` e `credentials` le modalità con cui ci si dovrà collegare a quest'ultimo. Si ricordi infatti che la replicazione con *syncrepl* avviene nella forma di una query LDAP, e dato che sarà necessario

[116] essendo possibili replicazioni *multi-master* si possono avere nella stessa configurazione più istanze della direttiva `syncrepl` per tutti quei casi in cui il *consumer* ottiene i dati da più *provider*; ciascuna di queste deve avere un diverso `rid`.

Argomento	Significato
`rid`	indica il numero identificativo (composto da tre cifre decimali) associato all'istanza di replicazione; deve essere specificato ad un valore diverso per ogni utilizzo della direttiva `syncrepl`.[116]
`provider`	indica la URI del *provider*, con la stessa sintassi illustrata in tab. 1.1, deve essere sempre specificato.
`type`	indica la modalità di sincronizzazione da usare: può assumere solo i due valori `refreshOnly` o `refreshAndPersist`.
`retry`	controlla i tentativi di riconnessione che devono essere eseguiti qualora la replicazione fallisca; prevede almeno una coppia di argomenti numerici indicanti rispettivamente l'intervallo fra un tentativo e l'altro (in secondi) ed il numero di ripetizioni (il carattere `+` indica un numero infinito), possono essere indicate più coppie di valori (in genere con intervalli crescenti) per specificare come continuare le riconnessioni una volta esauriti i tentativi indicati in una coppia precedente (il default è ripetere indefinitamente le riconnessioni ogni ora).
`searchbase`	indica la base della sezione di albero che si vuole replicare (è la base delle ricerche usate per il protocollo di sincronizzazione), deve essere sempre specificato.
`filter`	indica un eventuale filtro di ricerca che consente di selezionare quali dati si vogliono replicare, il default è `(objectclass=*)` per indicare una ricerca generica.
`scope`	indica la profondità della ricerca, il default è `sub`.
`schemachecking`	indica se è attivo il controllo di consistenza dei dati, di default è `off` (dato che in caso di replicazione parziale si possono avere inconsistenze temporanee negli attributi).
`binddn`	indica (come DN) l'utente con il quale effettuare il collegamento al *producer*; ci si accerti che questo abbia privilegi sufficienti per poter ottenere i dati.
`bindmethod`	indica la modalità con cui effettuare il collegamento al *producer*, può assumere i valori `simple` o `sasl`.
`credentials`	indica la password dell'utente con cui effettuare il collegamento.
`attrs`	indica gli attributi da replicare, se non indicato verranno presi tutti gli attributi definiti (il valore di default è `*,+` che indica sia gli attributi ordinari che quelli interni).

Tabella 1.29: Principali argomenti della direttiva `syncrepl`.

accedere a tutti dati del server che fa da *provider*, occorrerà utilizzare per l'accesso un utente dello stesso che abbia sufficienti privilegi di accesso (almeno la lettura) per tutto quanto si vuole replicare. Si tenga presente inoltre che la trasmissione dei dati viene fatta in chiaro, pertanto è comunque opportuno usare una connessione cifrata con SSL.

Il terzo argomento necessario è `searchbase` che indica la base delle richieste dei dati che verranno effettuate sul *provider*, a questo si possono aggiungere come ulteriori parametri per la ricerca gli argomenti `scope`, `filter`, `attrs`, che indicano rispettivamente la profondità della stessa, un eventuale filtro (con la sintassi di sez. 1.2.5) e la lista (separata da virgole) degli attributi richiesti. Questi possono non essere specificati, nel qual caso varranno i valori di default, che comportano una ricerca generica su tutto l'albero e per qualunque attributo al di sotto di `searchbase`.

Infine è sempre opportuno specificare le modalità con cui effettuare la replicazione tramite l'argomento `type`; questo può assumere uno dei due valori `refreshOnly` e `refreshAndPersist`, nel primo caso è compito del *consumer* ricontattare il *provider* ad intervalli regolari per farsi consegnare gli aggiornamenti (il default è un giorno ma può essere modificato con l'argomento `interval`), nel secondo caso una volta stabilita la connessione iniziale da parte del *consumer* è compito del *provider* fornire i successivi aggiornamenti.

Pertanto un esempio di un possibile configurazione di `slapd` per effettuare la replicazione sul server secondario (cioè sul *consumer*) facendo riferimento al *provider* configurato in precedenza,

potrebbe essere la seguente:

```
rootdn cn=admin_cons,dc=truelite,dc=it
index entryCSN,entryUUID eq

syncrepl rid=123
    provider=ldaps://master.truelite.it
    type=refreshAndPersist
    searchbase="dc=truelite,dc=it"
    filter="(objectClass=organizationalPerson)"
    scope=sub
    schemachecking=off
    bindmethod=simple
    binddn="cn=repl_user_prov,dc=truelite,dc=it"
    credentials=password_segretissima
```

1.4.4 Utilizzo avanzato di *syncrepl*

Nella precedente sezione abbiamo esaminato l'uso più elementare del meccanismo di *syncrepl* per realizzare una replicazione dei dati di un albero LDAP, in cui il *consumer* si collega al *provider*. Benché questa replicazione possa essere effettuata sia nelle due modalità *refreshOnly* e *refreshAndPersist*, in entrambi i casi deve comunque essere il server che fa da *consumer* a contattare inizialmente il server che fa da *provider*. Come accennato questo comporta un problema quando fra i due server è presente un firewall che non consente questo tipo di connessioni, come può avvenire quando il *consumer* è su una rete esterna ed il *provider* su una rete interna.

Con la replicazione classica effettuata con `slurpd` questo problema non si sussisteva in quanto era comunque il server *master* a contattare lo *slave* inviando i relativi cambiamenti, ma con l'uso di *syncrepl* appena illustrato questo non avviene e ci si trova nella necessità, in caso di presenza di un firewall di protezione, di dover aprire un accesso dall'esterno all'interno, cosa che non è detto sia sempre compatibile con le politiche di sicurezza.

Dato che a partire dalla versione 2.4 di OpenLDAP il supporto per `slurpd` è stato eliminato, per ovviare a questo problema è possibile utilizzare una diversa configurazione, chiamata *proxy replication*, in cui la replicazione avviene sempre a partire da connessioni originanti dal server *master*. In questo caso il risultato può essere ottenuto utilizzando sul server *master* una seconda istanza dell'albero che si appoggi però al *backend* `ldap`; sarà questa ad agire come *consumer* rispetto all'istanza principale ottenendo gli aggiornamenti tramite *syncrepl*; questi poi verranno inviati al server *slave* secondo lo schema illustrato in fig. 1.8 utilizzando le funzionalità di *proxy* fornite dal *backend* `ldap`.

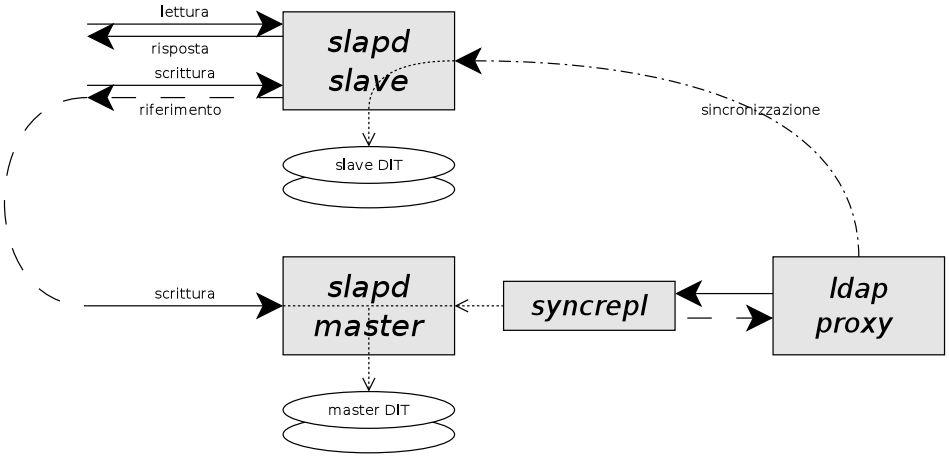


Figura 1.8: Il funzionamento della *proxy replication*.

In questo caso primo passo è quello di abilitare sul server *master* sia il modulo di gestione di *syncrepl* che quello del *backend* LDAP che utilizzeremo per fare da *proxy*, questo si fa aggiungendo con le direttive:

```
moduleload     syncrepl
moduleload     back_ldap
```

alla sezione generale del server *master*.

Il secondo passo è abilitare un ulteriore database che usi il *backend* `ldap` per fare da *proxy*. Questo database dovrà avere lo stesso suffisso usato per il database effettivo, ed accedervi con l'utente di amministrazione. Inoltre occorrerà configurarlo come *consumer* per *syncrepl* in maniera analoga a come fatto in sez. 1.4.3 per un server *slave* distinto, le direttive per fare tutto ciò sono le seguenti:

```
database        ldap
hidden          on
lastmod         on
suffix          "dc=truelite,dc=it"
restrict        all

# how to access to slave (where we proxy things)
# here must use slave user/base/credentials
rootdn          "cn=admin,dc=truelite,dc=it"
uri             ldaps://consumer.truelite.it
acl-bind        bindmethod=simple
                binddn="cn=admin,dc=truelite,dc=it"
                credentials=slave_secret

# how we get thing from the syncrepl overlay
# here must use master user/base/credentials
overlay         syncprov
syncrepl        rid=001
                provider=ldap://127.0.0.1
                binddn="cn=admin,dc=truelite,dc=it"
                bindmethod=simple
                credentials=master_secret
                searchbase="dc=truelite,dc=it"
                type=refreshAndPersist
                retry="5 5 300 5"
```

Fatto questo sarà sufficiente rendere edotto il lato *slave* di comportarsi come replica, questo si fa in maniera identica a come si faceva con `slurpd`, basterà cioè aggiungere le direttive:

```
updatedn   cn=admin,dc=truelite,dc=it
updateref  uri=ldaps://master.truelite.it:636
```

1.5 La centralizzazione della gestione di utenti e gruppi

Affronteremo in questa sezione le tematiche relative all'utilizzo di LDAP per la gestione centralizzata di utenti e gruppi. In realtà tratteremo più genericamente le modalità con cui è possibile gestire le informazioni di autenticazione (attraverso l'uso di PAM) e tutte le altre informazioni che di norma sono gestite dal *Name Service Switch* (password, hosts, gruppi, ecc.) appoggiandosi ad un server LDAP.

1.5.1 La creazione della infrastruttura dei dati

Per poter effettuare la gestione di utenti e gruppi di un sistema Unix su LDAP è necessario disporre delle *objectClass* che definiscono gli attributi necessari a mantenere tutte le informazioni

ad essi relative.[117] Queste informazioni sono state standardizzate da una serie di RFC,[118] e sono definite nel file `nis.schema`. In genere le *objectClass* necessarie (comprese quelle per le altre informazioni del NSS) vengono definite con la configurazione di default di `slapd.conf`, per cui normalmente non è necessaria nessuna riconfigurazione del server.

Attributo	Significato
`uid`	nome utente, equivalente del nome di login, è l'attributo principale usato anche nei DN che identificano le voci degli utenti; equivalente al primo campo di `/etc/passwd`.
`uidNumber`	valore numerico dell'*user-id* dell'utente; equivalente al terzo campo di `/etc/passwd`.
`gidNumber`	valore numerico del *group-id* del gruppo primario dell'utente; equivalente al quarto campo di `/etc/passwd`.
`gecos`	informazioni varie sull'utente; equivalente al quinto campo di `/etc/passwd`.
`homeDirectory`	home directory dell'utente; equivalente al sesto campo di `/etc/passwd`.
`loginShell`	shell di login dell'utente; equivalente al settimo campo di `/etc/passwd`.
`userPassword`	password cifrata dell'utente.[119]
`shadowLastChange`	numero del giorno, contato a partire dal 1 gennaio 1970, in cui la password è stata cambiata l'ultima volta; equivalente al terzo campo di `/etc/shadow`.
`shadowMin`	numero di giorni dall'ultimo cambiamento prima dei quali la password non può essere cambiata di nuovo; equivalente al quarto campo di `/etc/shadow`.
`shadowMax`	numero di giorni dall'ultimo cambiamento, dopo i quali la password scade e deve essere rinnovata; equivalente al quinto campo di `/etc/shadow`.
`shadowWarning`	numero di giorni prima della scadenza della password durante i quali l'utente viene avvertito; equivalente al sesto campo di `/etc/shadow`.
`shadowInactive`	numero di giorni dopo i quali, una volta scaduta la password senza rinnovo, l'account viene disabilitato; equivalente al settimo campo di `/etc/shadow`.
`shadowExpire`	numero del giorno, contato dal 1 gennaio 1970, in cui l'account viene disabilitato; equivalente all'ottavo campo di `/etc/shadow`.
`shadowFlag`	valore riservato, al momento non è utilizzato; equivalente al nono campo di `/etc/shadow`.

Tabella 1.30: Gli attributi contenenti i dati degli utenti, definiti nelle *objectClass* `posixAccount` e `shadowAccount`.

Per gli utenti le *objectClass* in questione sono `posixAccount` che contiene i dati classici che normalmente vengono mantenuti in `/etc/passwd` e `shadowAccount` che contiene i dati relativi alle estensioni delle *Shadow password*, usualmente mantenuti in `/etc/shadow`; nelle due sezioni di tab. 1.30 si sono riportati i relativi attributi, con l'indicazione dell'informazione in essi contenuta.

Oltre alle informazioni relative agli utenti, occorre ovviamente poter gestire anche quelle relative ai gruppi, per questo è viene utilizzata la *objectClass* `posixGroup`, che contiene i dati che classicamente verrebbero mantenuti in `/etc/group`;[120] in tab. 1.31 si sono riportati i vari attributi definiti in detta *objectClass* ed il relativo significato.

Come per qualunque altro dato è possibile utilizzare un client LDAP generico per inserire le informazioni relative ad utenti e gruppi direttamente nell'albero, andando a modificare gli

[117] sono le stesse che normalmente stanno nei classici file `/etc/passwd`, `/etc/group`, ecc. e che sono illustrate in dettaglio in sez. 4.3.3 di [AGL].

[118] per essere precisi gli RFC 2307 e RFC 2252.

[119] si è riportato anche questo attributo, che essendo definito in maniera più generale non è specifico né di `posixAccount` né di `shadowAccount`, in quanto contiene una delle informazioni essenziali per la gestione degli utenti.

[120] vengono mantenute solo le informazioni di base, non le estensioni relative a `/etc/gshadow`.

[121] anche in questo caso vale quanto detto per l'analogo attributo di `posixAccount`.

Attributo	Significato
`cn`	nome del gruppo; equivalente al primo campo di `/etc/group`.
`gidNumber`	valore numerico del *group-id* del gruppo; equivalente al terzo campo di `/etc/group`.
`memberUid`	username di un utente del gruppo, viene ripetuto per ciascun utente membro; equivalente ad un nome della lista mantenuta nel quarto campo di `/etc/group`.
`userPassword`	password cifrata del gruppo.[121]

Tabella 1.31: Gli attributi contenenti i dati dei gruppi, definiti nella *objectClass* `posixGroup`.

attributi elencati in tab. 1.30 e tab. 1.31; fare questa operazione manualmente è però un lavoro noioso e facilmente soggetto ad errori (molti dati, come ad esempio gli identificativi, sono correlati fra loro), per cui spesso si utilizzano dei programmi espressamente dedicati a questo scopo, che semplificano il lavoro e consentono di ricavare automaticamente le informazioni necessarie correlando i vari dati.

Uno dei pacchetti usati più spesso è quello dei `migrationtools`, un insieme di script che consentono sia di generare lo schema di un'infrastruttura su cui inserire le informazioni tipiche del NSS, che di migrare su LDAP i dati presenti nei vari file `/etc/passwd`, `/etc/shadow`, `/etc/hosts`, ecc. Per ciascuno di questi file gli script producono un corrispondente file `.ldif` contenente le stesse informazioni, che poi potranno essere inserite direttamente sul server LDAP con `ldapadd`.

Prima di poter utilizzare gli script di migrazione occorre specificare nel loro file di configurazione (su Debian è accessibile sotto `/etc/migrationtools/migrate_common.ph`) le impostazioni che indicano la sezione di albero in cui andranno inseriti i dati. Per far questo dovranno essere adattate al proprio caso le righe seguenti:

```
# Default DNS domain
$DEFAULT_MAIL_DOMAIN = "truelite.it";

# Default base
$DEFAULT_BASE = "dc=truelite,dc=it";
```

dopo di che il file `migrate_common.ph` deve essere copiato in `/usr/share/migrationtools/`, dove sono installati gli script, o essere referenziato con un link simbolico.[122] Per usare gli script di conversione ci si dovrà porre in detta directory.

Il primo passo è sempre quello di generare l'infrastruttura di base, lo script `migrate_base.pl` permette di creare un file `base.ldif` che contiene quanto necessario; il comando è:

```
[root@havnor migrationtools]# ./migrate_base.pl > base.ldif
```

In genere l'infrastruttura sui cui si mantengono i dati è costituita da una serie di `organizationalUnit` che devono essere inserite nell'albero per fare da radice delle varie sezioni al cui interno inseriremo i dati relativi ai servizi NSS che possono essere spostati su LDAP. Un esempio di una queste voci, estratto dal file `base.ldif` generato dal comando precedente, è il seguente:

```
dn: ou=Hosts,dc=truelite,dc=it
ou: Hosts
objectClass: top
objectClass: organizationalUnit
```

che definisce una `organizationalUnit` per fare da radice alla sezione di albero su cui poi verrà inserito l'elenco degli *host* mantenuti su LDAP che può essere aggiunto a quello di `/etc/hosts`.

[122] il pacchetto Debian esegue questa installazione di default.

Si tenga presente che alcune delle voci generate da `migrate_base.pl` possono essere superflue, inoltre sono previste delle voci anche per i dati di NIS, che certamente non interessano in quanto esso verrà sostituito appunto da LDAP. Infine alcune voci possono essere duplicate (come la base dell'albero) impedendo il funzionamento del comando di inserimento.[123] Per questo conviene comunque modificare opportunamente file `base.ldif` prodotto dal comando, per adattarlo ai propri scopi. Dopo di che si potranno inserire le nuove voci nell'elenco con qualcosa del tipo:

```
ldapadd -x -D"cn=admin,dc=truelite,dc=it" -W -f base.ldif
```

e si noti come si sia specificato un collegamento con il DN che ha i privilegi di amministrazione (sarà anche richiesta la relativa password) per poter scrivere sull'albero.

Un servizio che è possibile migrare su LDAP per fare una prova è quello della risoluzione statica dei nomi delle macchine usualmente fatta tramite il file `/etc/hosts`. Questo è un modo per mantenere gli indirizzi delle macchine di una rete locale che vengono modificati spesso, senza dover stare a configurare un DNS per un dominio fittizio. Il relativo script di migrazione è `migrate_hosts.pl`, che si esegue con il comando:

```
[root@havnor migrationtools]# ./migrate_hosts.pl /etc/hosts hosts.ldif
```

questo genererà il file `hosts.ldif` con le voci relative a ciascuna macchina già presente in `/etc/hosts`; una voce del tipo di:

```
192.168.1.152  roke.truelite.it      roke
```

sarà trasformata in:[124]

```
dn: cn=roke.truelite.it,ou=Hosts,dc=truelite,dc=it
objectClass: top
objectClass: ipHost
objectClass: device
ipHostNumber: 192.168.1.152
cn: roke.truelite.it
cn: roke
```

Dato che in `/etc/hosts` sono normalmente presenti anche gli indirizzi multicast di IPv6 ed il `localhost`, installati di default con la distribuzione, conviene di nuovo modificare a mano il file per togliere questi e tutti gli eventuali indirizzi che non si vogliono inserire nell'elenco; inoltre conviene rimuovere anche tutte le voci duplicate. Effettuata la pulizia si potranno di nuovo immettere i dati con il comando:

```
[piccardi@havnor ldap]$ ldapadd -x -D"cn=admin,dc=truelite,dc=it" -W -f hosts.ldif
```

ed a questo punto si potrà verificare che le informazioni immesse siano effettivamente disponibili,[125] usando il comando:

```
[piccardi@havnor ldap]$  ldapsearch -x -LL "(cn=roke)"
```

In questo modo, una volta configurato il servizio del *Name Service Switch* come illustrato in sez. 1.5.2, si potrà verificare che la risoluzione del nome funzioni[126] con:

[123] a meno di non invocare `ldapadd` con l'opzione `-c`.

[124] in questo caso viene utilizzata la *objectclass* `ipHost`, presente anch'essa all'interno di `nis.schema`, che definisce gli attributi necessari.

[125] prima sarà anche necessario eseguire le corrette impostazioni di `nsswitch.conf`, come illustrato in sez. 1.5.2.

[126] con alcune versioni di `libnss-ldap` la cosa sembra funzionare solo per programmi che usano la funzione `gethostbyname`, i programmi (come `ssh`, `telnet`, ecc.) che usano `getaddrinfo` non funzionano.

```
[piccardi@havnor piccardi]$ ping roke
PING roke.truelite.it (192.168.1.152): 56 data bytes
64 bytes from 192.168.1.152: icmp_seq=0 ttl=255 time=9.5 ms
64 bytes from 192.168.1.152: icmp_seq=1 ttl=255 time=11.4 ms
64 bytes from 192.168.1.152: icmp_seq=2 ttl=255 time=8.0 ms
64 bytes from 192.168.1.152: icmp_seq=3 ttl=255 time=8.2 ms

--- roke.truelite.it ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 8.0/9.2/11.4 ms
```

Questo primo passo può essere fatto come esercizio per verificare se si è capito bene il funzionamento del meccanismo, sul piano pratico è in genere molto più semplice definire una zona locale sul DNS. Le informazioni che invece è sicuramente interessante portare su LDAP sono quelle relativi agli utenti e ai gruppi, così da centralizzarne la gestione.

In questo caso il primo passo è inserire i dati dei gruppi. Al solito si usano gli script del pacchetto `migrationtools`. Con lo script `migrate_group.pl` si effettua la conversione in formato LDIF dei dati di `/etc/group`; il comando da utilizzare è:

```
[root@havnor migrationtools]# ./migrate_group.pl /etc/group group.ldif
```

che crea le voci relative a tutti i gruppi presenti in `/etc/group` in `group.ldif`.

Il problema è che in `/etc/group` sono presenti anche tutti i gruppi di sistema (come `cdrom`, `audio`, ecc.) che non ha molto senso spostare su LDAP, poiché sono collegati a quanto è stato installato sulla singola macchina, e che per questo possono anche essere diversi da una installazione ad un'altra. Per questo motivo è opportuno modificare il file lasciando solo le voci relative ai gruppi effettivamente utilizzati nella gestione degli utenti. Così per i gruppi principali degli utenti avremo qualcosa delle voci del tipo:

```
dn: cn=piccardi,ou=Group,dc=truelite,dc=it
objectClass: posixGroup
objectClass: top
cn: piccardi
userPassword: {crypt}x
gidNumber: 1000
```

mentre se abbiamo dei gruppi utilizzati come gruppi di lavoro avremo qualcosa del tipo di:

```
dn: cn=cvsusers,ou=Group,dc=truelite,dc=it
objectClass: posixGroup
objectClass: top
cn: cvsusers
userPassword: {crypt}x
gidNumber: 103
memberUid: simone
memberUid: chris
memberUid: csurchi
```

Una volta migrati i gruppi si potrà ripetere il procedimento con gli utenti, in questo caso lo script da utilizzare è `migrate_passwd.pl`, ed il comando usato è:

```
[root@havnor migrationtools]# ./migrate_passwd.pl /etc/passwd passwd.ldif
```

che migrerà sia le informazioni presenti in `/etc/passwd` che quelle presenti in `/etc/shadow`.

Anche in questo caso occorrerà provvedere ad eliminare dall'elenco tutti gli utenti di sistema (come `root`, `daemon`, ecc.) che di norma sono locali alle varie macchine e non devono (in particolar modo `root`) essere messi su LDAP. Un esempio di voce relativa ad un utente reale è la seguente:

```
dn: uid=piccardi,ou=People,dc=truelite,dc=it
uid: piccardi
cn::U2ltb25lIFBpY2NhcmRp
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: {crypt}$1$7IDw9Tn0$/QvjoR4CNuQptwLmtGIQA0
shadowLastChange: 11354
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1002
gidNumber: 1002
homeDirectory: /home/piccardi
gecos: Simone Piccardi,,,
```

Una volta ripuliti entrambi i file generati con gli script di migrazione si potrà passare all'inserimento dei dati nell'elenco, questo si fa con i comandi:

```
ldapadd -x -D"cn=admin,dc=truelite,dc=it" -W -f group.ldif
ldapadd -x -D"cn=admin,dc=truelite,dc=it" -W -f passwd.ldif
```

Una volta inseriti i dati iniziali resta il problema di come creare o cancellare gli utenti quando questi sono su LDAP. Infatti benché alcune delle informazioni relative agli utenti siano gestite in maniera trasparente da PAM,[127] questo non avviene per la creazione e rimozione di utenti e gruppi, in quando i comandi classici `useradd` e `userdel` (e gli analoghi per i gruppi illustrati in sez. 4.3.2 di [AGL]) operano esclusivamente sui file `/etc/passwd` ed `/etc/shadow`.

Un programma che consente di ottenere le stesse funzionalità di questi programmi, ma che è in grado di operare anche quando gli utenti sono su LDAP, è `cpu` (su Debian è installabile con il pacchetto omonimo). Il comando replica le funzionalità dei comandi classici Unix per la creazione di utenti e gruppi,[128] utilizzando anche la stessa sintassi; basterà cioè scrivere dopo `cpu` il comando che si sarebbe utilizzato normalmente, perché l'operazione venga eseguita su LDAP.

Per poter utilizzare `cpu` occorre però, come per i `migrationtools`, indicare su quale sezione dell'albero sono mantenuti i dati degli utenti. Inoltre il programma opera direttamente sull'albero usando il protocollo, per cui occorre anche definire un utente per conto del quale collegarsi al server, che ovviamente deve essere in grado di modificare i dati presenti su di esso. Tutto questo viene fatto nel file di configurazione `/etc/cpu/cpu.conf`, che usa il formato dei file `.ini` di Windows.

Il file prevede al momento due sole sezioni,[129] quella che contiene le configurazioni generali, identificata dalla parola chiave `GLOBAL`, e quella che contiene le configurazioni di LDAP, identificata dalla parola chiave `LDAP`. La sezione generale supporta due sole opzioni, `DEFAULT_METHOD`, che indica il metodo di amministrazione degli utenti (e può solo assumere i valori `ldap` o `passwd`) e `CRACKLIB_DICTIONARY` che permette di indicare un file con il dizionario delle password da controllare, qualora sia abilitato il supporto per le *cracklib*.[130]

[127] ad esempio il cambio password, come vedremo in sez. 1.5.3.

[128] cioè `useradd`, `userdel`, `usermod`, e gli analoghi per i gruppi, ma non le funzioni delle versioni "*semplificate*" come `adduser` e `deluser`.

[129] essendo `cpu` sviluppato modularmente è previsto che esso possa operare utilizzando diversi tipi di supporto per i dati degli utenti, ciascuno dei quali ha una sua sezione di configurazione specifica; al momento gli unici supporti indicati sono però LDAP e i file classici di Unix, e questi ultimi non sono supportati, per cui si usa solo la sezione globale e quella relativa a LDAP.

[130] le *cracklib* sono costituite da una libreria ed un insieme di programmi che consentono di verificare la *forza* di una password, intesa come capacità di resistenza ad un tentativo di *attacco a dizionario* (una trattazione di queste problematiche è in sez. 1.2.1 di [SGL].

Opzione	Significato
DEFAULT_METHOD	metodo di autenticazione; può assumere soltanto i valori ldap e passwd.
CRACKLIB_DICTIONARY	indica il file (con il pathname) del dizionario delle password per le *cracklib*.
LDAP_URI	la URI del server LDAP su cui sono mantenute le informazioni di utenti e gruppi.
BIND_DN	il *Distinguished Name* dell'utente di LDAP per conto del quale collegarsi al server.
BIND_PASS	la password del precedente utente (in chiaro).
USER_BASE	la sezione di albero (la base della ricerca) sotto la quale sono mantenuti i dati degli utenti.
GROUP_BASE	la sezione di albero (la base della ricerca) sotto la quale sono mantenuti i dati dei gruppi.
DEFAULT_SHELL	la shell di default assegnata all'utente.

Tabella 1.32: Principali opzioni di configurazione usate da `cpu.conf`.

La sezione di configurazione principale è pertanto quella relativa ai parametri di LDAP, i principali dei quali sono riportati nella seconda parte di tab. 1.32. In particolare si dovrà specificare con `LDAP_URI` il server a cui rivolgersi, con `BIND_DN` il *Distinguished Name* l'utente con cui collegarsi al server e con `BIND_PASS` la password di detto utente; per questo motivo (dato che la password è mantenuta in chiaro) è importante verificare che il file non sia leggibile. Le sezioni di albero su cui si trovano i dati di utenti e gruppi dovranno essere indicate rispettivamente con `USER_BASE` e `GROUP_BASE` in forma di basi di ricerca.

I principali parametri da modificare rispetto al file installato con il pacchetto sono illustrati dal seguente esempio; tutti gli altri possono essere lasciati ai valori di default:

```
[GLOBAL]
DEFAULT_METHOD  = ldap
#CRACKLIB_DICTIONARY = /var/cache/cracklib/cracklib_dict

[LDAP]
LDAP_URI                = ldap://localhost
BIND_DN                 = cn=admin,dc=truelite,dc=it
BIND_PASS               = passwordsegretissima
USER_BASE               = ou=People,dc=truelite,dc=it
GROUP_BASE              = ou=Group,dc=truelite,dc=it
```

1.5.2 La configurazione del *Name Service Switch* su LDAP

Per utilizzare il supporto di LDAP per il *Name Service Switch*[131] basterà installare il pacchetto `libnss-ldap`, che fornisce la relativa libreria dinamica. I dati di configurazione per l'accesso al server LDAP sono mantenuti nel file `/etc/libnss-ldap.conf`,[132] che ha la stessa sintassi di `ldap.conf`, solo che in questo caso, oltre alle direttive già viste in tab. 1.6 (che mantengono lo stesso significato) sono disponibili alcune direttive aggiuntive, riconosciute solo per questo file. Le più significative sono illustrate in tab. 1.33, al solito l'elenco completo è disponibile nella pagina di manuale, accessibile con `man libnss-ldap.conf`.

In genere le direttive di tab. 1.33 non vengono usate esplicitamente, essendo i loro valori di default validi per la maggior parte degli usi; restano pertanto solo le direttive di base, e, se si utilizzano connessioni su TLS/SSL (come è sempre opportuno fare) occorrerà anche ricorrere

[131] daremo per acquisita una familiarità di base con i concetti del *Name Service Switch*; in caso contrario una trattazione degli stessi è disponibile in sez.3.1.3 di [AGL].

[132] questo vale per le ultime versioni di Debian, in alcune vecchie versioni di RedHat viene usato invece `/etc/ldap.conf`; per sapere quale è il file che viene usato nella vostra distribuzione potete guardare all'output di un comando come: `strings /lib/libnss_ldap.so.2 | grep /etc`.

Direttiva	Significato
`bind_policy`	imposta la politica di collegamento al server ed il comportamento in caso di fallimento della connessione; prende i valori `hard_open`, `hard_init` e `soft`.
`nss_connect_policy`	imposta la politica per le connessioni delle ricerche, prende i valori `persist`, che mantiene aperta la connessione (il default) e `oneshot` che la chiude esaurita la ricerca.
`idle_timelimit`	in caso di politica `connect` imposta un valore di timeout (in secondi) oltre il quale chiude la connessione se non ci sono richieste (il default è zero, che mantiene sempre la connessione attiva).
`nss_base_<map>`	specifica la base di ricerca per i vari servizi del NSS (dove `<map>` può assumere i valori `passwd`, `shadow`, `group`, ecc.) e prende come argomento il DN della base della ricerca nella forma `basedn?scope?filter`.
`nss_map_attribute`	consente di rimappare la richiesta di un certo attributo su un altro; prende come argomenti il nome dell'attributo da rimappare seguito da quello su cui rimapparlo.
`nss_map_objectclass`	consente di rimappare la richiesta di una *objectClass*; prende come argomenti il nome dell'*objectClass* da rimappare, seguito da quella su cui rimapparla.

Tabella 1.33: Direttive di configurazione specifiche di `libnss-ldap.conf` riconosciute anche da `pam_ldap.conf`.

alle direttive illustrate in tab. 1.24 come fatto per `ldap.conf`. Un esempio di `/etc/libnss-ldap.conf` è il seguente:

```
BASE            dc=truelite,dc=it
URI             ldaps://ldap.truelite.it/
TLS_CHECKPEER   yes
TLS_CACERT      /etc/ssl/certs/cacert.pem
rootbinddn      cn=admin,dc=truelite,dc=it
```

In questo caso si usa la direttiva `URI` che permette di specificare il server LDAP su cui sono mantenuti i dati, mentre tramite con la direttiva `BASE` si indica la base delle ricerche all'interno dell'albero[133] per ottenere i dati relativi agli utenti. Seguono le direttive già viste in sez. 1.3.2 per attivare la connessione cifrata con SSL; ovviamente si dovrà aver cura di copiare il certificato `cacert.pem` su tutte le macchine che centralizzeranno NSS su LDAP.

Infine con `rootbinddn` si specifica il DN dell'utente LDAP con il quale ci si vuole autenticare presso il server quando l'utente (quello Unix della macchina locale) per conto del quale si esegue un comando che utilizza le funzioni del *Name Service Switch* è l'amministratore. Come accennato in tab. 1.6, la password associata a questo utente deve essere specificata su una riga singola nel file `/etc/ldap.secret`; trattandosi della password che da l'accesso completo ai dati degli utenti sul server LDAP questo file deve essere opportunamente protetto da lettura.[134] Questo è necessario per garantire all'amministratore di una macchina di avere sugli utenti mantenuti su LDAP gli stessi privilegi che avrebbe sugli utenti locali, in particolare quello di poter operare sui campi relativi alle password.

In generale non è mai una buona politica garantire questo privilegio alle macchine che usano il server per l'autenticazione centralizzata, dato che su di esse non si può avere un controllo diretto, per cui chiunque abbia accesso fisico alle macchine stesse potrebbe facilmente leggere il file `/etc/ldap.secret`. Non ha senso infatti che l'amministratore di un client abbia le capacità di conoscere e modificare i dati relativi ad utenti presenti anche su altre macchine che non sono sotto il suo diretto controllo. L'unico caso in cui vale la pena usare questa configurazione è sul server LDAP stesso (che si suppone adeguatamente protetto rispetto l'accesso fisico); in tal caso

[133] nel caso è la stessa che abbiamo in `/etc/ldap/ldap.conf` e la si potrebbe omettere.

[134] è per poter mantenere `/etc/libnss-ldap.conf` accessibile in lettura che la password viene messa a parte su questo file.

infatti ha senso che l'amministratore possa avere gli stessi privilegi sia sugli utenti locali che su quelli mantenuti su LDAP.[135]

Normalmente la ricerca dei dati per il *Name Service Switch* viene effettuata su tutto l'albero,[136] richiedendo tutti gli attributi delle voci che appartengono alle *objectClass* necessarie;[137] esistono però dei casi in cui è necessario essere essere più specifici; ad esempio si può volere limitare la ricerca (anche solo per renderla più veloce) ad una certa sezione dell'albero.

Per questo si possono usare le varie direttive `nss_base_<map>` (dove `<map>` può essere `hosts`, `passwd`, `group` o uno qualunque dei servizi che vengono forniti dal *Name Service Switch*). Esse permettono, per ciascuna delle diverse classi di informazioni, di recuperare i relativi dati su diverse specifiche sezioni dell'albero. Così per indicare dove reperire le informazioni sugli hostname delle macchine, dei dati delle password, degli utenti e dei gruppi si potrà utilizzare qualcosa del tipo:

```
nss_base_hosts      ou=Hosts,dc=truelite,dc=it?one
nss_base_passwd     ou=People,dc=truelite,dc=it?one
nss_base_shadow     ou=People,dc=truelite,dc=it?one
nss_base_group      ou=Group,dc=truelite,dc=it?one
```

dove l'argomento è sempre nella forma di un DN seguito da un punto interrogativo e da un indicatore della profondità a cui effettuare la ricerca,[138] che assume uno dei valori fra quelli già illustrati in tab. 1.17.

Una volta impostato `libnss-ldap.conf`, che si limita a indicare dove recuperare le informazioni sul server, occorrerà indicare al *Name Service Switch* di utilizzare anche LDAP per eseguire le sue ricerche, questo va fatto modificando opportunamente il file `/etc/nsswitch.conf` per fargli prendere i dati da LDAP, inserendo al suo interno un contenuto del tipo:

```
passwd:         compat ldap
group:          compat ldap
shadow:         compat ldap
hosts:          files dns ldap
```

nel caso si utilizzi LDAP per risolvere i nomi delle macchine[139] si deve fare attenzione che la macchina su cui sta il server LDAP sia risolvibile senza l'uso di LDAP, (è pertanto escluso di utilizzare come primo supporto `ldap`) altrimenti si creerebbe un ciclo vizioso in cui `libnss-ldap` chiama `gethostbyname` per avere tale indirizzo, che a sua volta richiama di nuovo `libnss-ldap` per interrogare LDAP, con la conseguenza di generare un segmentation fault per tutti i programmi che usano tale funzione.

A questo punto si può verificare che le informazioni di utenti e gruppi vengano viste correttamente, per questo si può usare il comando `getent` che interroga il *Name Service Switch*. Per verificare che gli utenti siano correttamente rilevati si potrà usare il comando:

```
[root@havnor libpam-ldap]# getent passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
...
piccardi:x:1002:1002:Simone Piccardi,,,:/home/piccardi:/bin/bash
admin:x:1003:1003:Utente Amministrativo,,,:/home/admin:/bin/bash
cgabriel:x:1004:1004:Christopher R. Gabriel,,,:/home/cgabriel:/bin/bash
csurchi:x:1005:1005:Christian Surchi,,,:/home/csurchi:/bin/bash
...
```

[135] nel caso, ovviamente, che anche sul server si stia usando LDAP per mantenere i dati degli utenti.

[136] a partire dalla base impostata con la direttiva `BASE`.

[137] il pacchetto `libnss-ldap` contiene il supporto per la ricerca delle informazioni sulle *objectClass* definite nell'RFC 2307.

[138] ed eventualmente da un altro punto interrogativo e da un filtro di ricerca, anche se questo non è molto comune.

[139] soluzione sconsigliata, è molto più semplice predisporre una zona locale su un server DNS ad uso interno.

mentre per verificare che sia corretta l'impostazione dei gruppi si potrà usare il comando:

```
[root@havnor libpam-ldap]# getent group
root:x:0:
daemon:x:1:
bin:x:2:
...
piccardi:x:1002:
admin:x:1003:
cgabriel:x:1004:
csurchi:x:1005:
...
```

infine per verificare anche le altre impostazioni sulle *Shadow password* si potrà eseguire il comando:

```
[root@havnor libpam-ldap]# getent shadow
root:$1$jjGU2TK$jHJVQBp/Xb6bmUBzmT7ubjY:11376:0:99999:7:::
daemon:*:11376:0:99999:7:::
bin:*:11376:0:99999:7:::
...
piccardi:x:12271::99999:7:::0
admin:x:12261::99999:7:::0
cgabriel:x:12289::99999:7:::0
csurchi:x:12289::99999:7:::0
...
```

Si noti come il risultato per gli utenti ed i gruppi appena ottenuto sia identico a quello che si avrebbe con la configurazione classica,[140] mentre quello delle *shadow password* non restituisce nessun valore per l'hash della password per tutti gli utenti che abbiamo messo su LDAP. Questo è corretto perché da un client non si deve avere un accesso privilegiato, ed è quindi impossibile leggere il campo che contiene le password. Per poter permettere all'amministratore di leggere le password degli utenti si deve usare la direttiva `rootbinddn`, ed in tal caso si otterrà un risultato del tipo:

```
...
piccardi:$1$YhWNGGxB$nGFYcKX3g/Ep3NkNSJUGT1:12271::99999:7:::0
admin:$1$VwgNDkZj$mpT4ZN452yjMGqioqN/Ej0:12261::99999:7:::0
cgabriel:shUXAXmXNOs/Y:12289::99999:7:::0
csurchi:5.1Fn2ynoiIqs:12289::99999:7:::0
...
```

che effettivamente contiene tutti i dati che ci aspettiamo.

1.5.3 La configurazione di PAM per l'autenticazione su LDAP

Una volta centralizzati gli identificativi su NSS, il passo successivo è quello di configurare PAM[141] per mantenere su LDAP le credenziali di utenti e gruppi. Una volta fatto questo si avrà un sistema di autenticazione e gestione degli utenti centralizzato in grado di sostituire il servizio NIS.

Rispetto a NIS infatti l'uso di PAM su LDAP presenta numerosi vantaggi: anzitutto l'informazione mantenuta nel database può essere utilizzata anche per altri scopi (come per la gestione

[140] una possibile anomalia potrebbe verificarsi con la presenza doppia di un utente, qualora esso sia stato inserito sia come utente locale che su LDAP, in casi come questo i programmi che si appoggiano al NSS usano sempre il valore restituito per primo (nel nostro caso l'utente locale).

[141] *Pluggable Authentication Method*, per una trattazione si veda sez.4.3.4 di [AGL].

degli account di posta o di altre informazioni generali) e le informazioni di autenticazione possono essere usate anche da altri programmi. Inoltre LDAP permette un controllo di accesso alle varie informazioni memorizzate nell'elenco molto più dettagliato attraverso l'uso delle *access list* trattate in sez. 1.3.1, infine supporta la replicazione e la ridondanza, ed è pertanto in grado di rispondere ad esigenze di resistenza agli incidenti. Il vantaggio più importante rispetto a NIS però è quello della sicurezza: LDAP può essere utilizzato attraverso TLS/SSL, consentendo la trasmissione cifrata dei dati, la autenticazione di client e server, e la verifica dell'integrità dei dati.

Il pacchetto che consente l'uso di PAM con LDAP è `libpam-ldap`, che fornisce la apposita libreria `libpam_ldap.so`. Il comportamento della libreria è controllato dal file di configurazione `/etc/pam_ldap.conf`.[142] Questo ha un formato sostanzialmente identico a quello di `libnss-ldap.conf`, e prevede, oltre alle direttive di base di tab. 1.6, anche le estensioni illustrate in tab. 1.33.

Sono inoltre previste ulteriori direttive specifiche, che permettono di impostare le caratteristiche di funzionamento proprie di questo modulo, come `pam_password` che indica il formato con cui la password deve essere memorizzata su LDAP, o `pam_min_uid` che può essere utile per restringere gli *user-id* degli utenti gestiti su LDAP.[143]

Un elenco delle principali direttive specifiche di `pam_ldap.conf` è riportato in tab. 1.34, l'elenco completo è nella pagina di manuale, accessibile con `man pam_ldap.conf`. Per la maggior parte di esse è corretto il valore di default e non è necessario utilizzarle esplicitamente.

Direttiva	Significato
`pam_password`	specifica il protocollo con cui eseguire la modifica di una password all'interno dell'albero di LDAP, può assumere i valori: `plain` per una semplice trasmissione in chiaro, `crypt` per una cifratura nel formato tradizionale di Unix, `exop` per il protocollo descritto nell'RFC 3062.
`pam_login_attribute`	indica l'attributo che deve essere utilizzato per la ricerca dello *username* dell'utente; il default è `uid`.
`pam_min_uid`	indica il numero minimo di user ID consentito.
`pam_max_uid`	indica il numero massimo di user ID consentito.

Tabella 1.34: Direttive di configurazione usate da `pam_ldap.conf`.

Con Debian all'installazione del pacchetto il sistema del *debconf* richiede i dati per collegarsi al server creando automaticamente uno scheletro di configurazione per `pam_ldap.conf`. Se si vuole utilizzare TLS/SSL, occorrerà inoltre impostare le direttive illustrate in sez. 1.3.2. Un esempio di questo file, eliminati commenti e righe vuote, è il seguente:

```
base dc=truelite,dc=it
uri ldaps://ldap.truelite.it/
tls_checkpeer yes
TLS_CACERT /etc/ssl/certs/cacert.pem
ldap_version 3
rootbinddn cn=admin,dc=truelite,dc=it
pam_password exop
```

Per gestire i dati di autenticazione può convenire creare un utente dedicato sull'elenco, da tenere separato dall'utente usato come amministratore per tutto l'elenco. Questo può permettere di avere un maggiore controllo su quanto i vari client potranno fare, limitando l'accesso alle sole informazioni che riguardano l'autenticazione. Spesso però, quando le informazioni di

[142] di nuovo questo è il default usato da Debian, sono possibili altre scelte; in caso di dubbi si può verificare quale file viene utilizzato con lo stesso metodo illustrato in sez. 1.5.2 per `libnss-ldap.conf`.

[143] ad esempio impostando `pam_min_uid 100` si fa sì che non si possano ottenere da LDAP gli utenti di sistema usati per i demoni, e lo stesso root, evitando che un'eventuale capacità di scrittura sull'albero consenta l'inserimento di utenti privilegiati.

autenticazione sono le più rilevanti fra quelle che si mantengono nell'albero, si preferisce adottare questa politica per le altre informazioni meno importanti in esso presenti, ed utilizzare per i dati di autenticazione direttamente l'amministratore del database.

Per l'uso di PAM hanno particolare rilevanza i privilegi di accesso per l'attributo `userPassword`, in quanto è questo che viene utilizzato per l'autenticazione degli utenti; infatti se riprendiamo l'estratto del file di configurazione standard installato da Debian, già illustrato in sez. 1.3.1, vediamo che questo è trattato in modo particolare:

```
access to attrs=userPassword
        by dn="cn=admin,dc=truelite,dc=it" write
        by anonymous auth
        by self write
        by * none
```

L'accesso in scrittura all'utente di amministrazione del server è equivalente a quello che ha *root* per le password normali, esso cioè può cambiare le password di tutti gli utenti. Se si volesse introdurre un utente di amministrazione da usare al posto dell'amministratore generale si dovrebbe inserire anche per lui un adeguato permesso di accesso.

Come nel caso del NSS si ripresenta il problema di come trattare le richieste fatte dall'amministratore delle varie macchine che si appoggiano al server LDAP per l'autenticazione. La direttiva `rootbinddn` consente di associare queste all'utente di amministrazione, ma si ha il solito problema che l'amministratore del client avrebbe la capacità di modificare le credenziali di autenticazione sul server. Come nel caso precedente questa impostazione ha senso solo sul server LDAP stesso.

Si tenga presente che perché un utente possa modificare la propria password non è necessario che il client abbia pieno accesso a LDAP, in quanto esso potrà farlo una volta che si è autenticato, avendo utilizzato la ACL `by self write`. Questo stesso privilegio dovrà essere assegnato per gli altri attributi che si vuole l'utente possa modificare; in particolare l'accesso a `shadowLastChange` è fondamentale perché all'aggiornamento della password questo dato venga aggiornato.[144] Per questo motivo una configurazione più adeguata delle ACL per gli attributi usati nell'autenticazione degli utenti potrebbe essere:

```
access to attrs=userPassword
        by dn="cn=admin,dc=truelite,dc=it" write
        by anonymous auth
        by self write
        by * none
access to attrs=shadowLastChange,loginShell,gecos
        by dn="cn=admin,dc=truelite,dc=it" write
        by self write
        by * none
```

Una volta configurate le librerie per l'accesso al server, per poter centralizzare l'autenticazione occorrerà anche configurare PAM per fargli utilizzare il modulo `pam_ldap.so` con i vari servizi. Nel caso di Debian (e di tutte le distribuzioni più recenti) la configurazione di PAM avviene attraverso una serie di file posti nella directory `/etc/pam.d`, uno per servizio, che indicano quali funzionalità di PAM devono essere utilizzate.[145]

Il metodo classico (da usare con le versioni più vecchie di PAM) prevede che per ciascun servizio che vuole usare LDAP si debbano aggiungere le seguenti righe:

```
auth       sufficient pam_ldap.so
account    sufficient pam_ldap.so
password   sufficient pam_ldap.so
```

[144] altrimenti la password verrebbe cambiata, ma per il meccanismo di verifica della durata il cambiamento non avrebbe effetto, non venendo aggiornato il valore di questo attributo.

[145] per una descrizione più dettagliata del funzionamento di PAM si veda sempre [AGL], sez. 4.3.4.

prima delle equivalenti che usano `pam_unix.so`. Questo consente di utilizzare le informazioni su LDAP, se ci sono, e altrimenti passare al meccanismo classico di autenticazione di Unix. Si noti che non si è specificato niente per `session`; infatti stiamo solo autenticando su LDAP, le informazioni sulla sessione non sono su LDAP e non vengono gestite da `libpam-ldap`.

Questo metodo ha però lo svantaggio di dover modificare tutti i file presenti in `/etc/pam.d`. Le versioni più recenti di PAM (quelle in uso in tutte le principali distribuzioni) consentono l'inclusione di alcuni valori comuni mantenuti su file a parte. Questo in Debian è realizzato tramite i 4 file `common-auth`, `common-session`, `common-account` e `common-password`, per cui sarà sufficiente effettuare le modifiche citate all'interno di questi.

La configurazione suggerita presenta comunque alcuni inconvenienti; il primo è che cominciando con `pam_ldap.so` si andrà sempre a cercare prima su LDAP e poi in locale, questo comporta del traffico inutile quando si cercano dei dati relativi agli utenti locali. Dato che la ricerca in locale è molto meno onerosa sarebbe preferibile utilizzare `pam_ldap.so` dopo l'uso del classico `pam_unix.so`. Il secondo problema è che tutte le volte che un utente non è presente su LDAP la password viene richiesta una seconda volta.

Per questo motivo una configurazione più corretta potrebbe essere la seguente; che riportiamo per ciascuno dei quattro file comuni appena citati; per `common-auth` si potranno utilizzare le seguenti impostazioni:

```
auth        required      /lib/security/pam_env.so
auth        sufficient    /lib/security/pam_unix.so likeauth nullok
auth        required      /lib/security/pam_ldap.so use_first_pass
```

in questo caso prima si esegue la richiesta in locale e solo in caso di fallimento si effettua la richiesta su LDAP; inoltre con l'uso dell'argomento `use_first_pass` la password immessa la prima volta per l'autenticazione classica viene immediatamente riutilizzata anche per LDAP, evitando così la ripetizione della richiesta. In maniera analoga si modificherà `common-password` con:

```
password    sufficient    /lib/security/pam_unix.so nullok md5 shadow
password    required      /lib/security/pam_ldap.so use_first_pass
```

infine per `common-account` si potrà utilizzare:

```
account sufficient  /lib/security/pam_ldap.so
account sufficient  /lib/security/pam_unix.so
```

mentre per quanto riguarda `common-session` (per il quale non vengono sostanzialmente utilizzate le informazioni di LDAP) si potrà usare:

```
session required    /lib/security/pam_unix.so
session optional    /lib/security/pam_ldap.so
```

Infine se si vogliono ottimizzare le prestazioni sarà necessario indicizzare opportunamente gli attributi usati per l'autenticazione. Un esempio di indicizzazione per la configurazione nell'uso come server di autenticazione è dato dal seguente estratto di `slapd.conf`:

```
index objectClass pres,eq
index cn,sn,uid,displayName pres,sub,eq
index uidNumber,gidNumber eq,pres
index memberUid,mail,givenname eq,subinitial
index uniqueMember pres
```

1.6 L'integrazione dei servizi su LDAP

Vedremo in questa sezione come si può utilizzare LDAP come supporto di integrazione per mantenere centralizzate informazioni utilizzate da una serie di altri servizi e programmi che non siano la gestione di utenti e gruppi di un insieme di macchine, già illustrata in sez. 1.5.

1.6.1 La gestione dell'indirizzario

L'utilizzo classico di LDAP, quello per cui il servizio è nato, è quello di permettere la gestione centralizzata e la distribuzione via rete di un indirizzario (o rubrica). Vedremo in questa sezione quali sono gli schemi da utilizzare a questo scopo, e come configurare i vari client che possono appoggiarsi ad LDAP per la gestione dell'indirizzario.

Per la gestione dei dati di una rubrica condivisa esiste già una standardizzazione che definisce una serie di classi e relativi attributi che consentono di memorizzare i dati classici (nome, cognome, numero di telefono, indirizzo fisico e di posta elettronica, ecc.) che si trovano solitamente in un indirizzario. Alcuni programmi poi (ad esempio *Evolution*) utilizzano delle estensioni per mantenere dati aggiuntivi ad uso del programma; in questo caso occorrerà utilizzare i relativi schemi (includendoli nella configurazione di `slapd`, si veda sez. 1.2.3) che definiscono le classi e gli attributi necessari.[146]

L'utilizzo di questi schemi aggiuntivi di norma è da sconsigliare, a meno di non avere fin dall'inizio la possibilità di imporre a lungo termine una forte uniformità nell'uso delle specifiche applicazioni che ne fanno uso. Questo perché quando si introducono classi ed attributi particolari, essi sono riconosciuti solo dall'applicazione che li usa, e l'interoperabilità con altre applicazioni diventa più complessa,[147] non essendo queste consapevoli della presenza di dati aggiuntivi, che in un eventuale passaggio diventerebbero inutilizzabili.

Lo schema classico usato per la gestione delle informazioni di un indirizzario è `inetorgperson.schema`, distribuito con gli *schema* di default di OpenLDAP; esso di solito viene incluso nella configurazione di default in `slapd.conf`. Questo schema contiene la definizione della *objectClass* `inetOrgPerson` relativa ad una persona, che definisce una lunga lista di attributi contenenti dati personali come il telefono, l'indirizzo di casa, quello di posta elettronica e similari.[148]

Se su un database si mantengono sia questi dati che quelli di un insieme di utenti centralizzati, è buona norma dedicare ai dati della rubrica una sezione dell'albero a parte[149] così da restringere le ricerche su quella sezione specifica di albero e non mescolare dati di minore rilevanza, come questi, coi dati degli utenti.

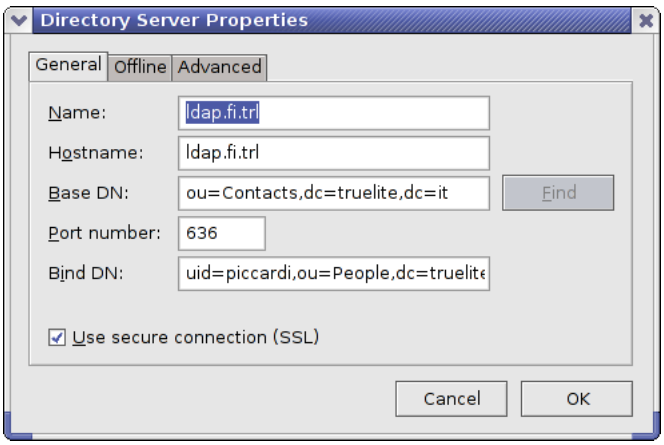


Figura 1.9: Schermata di configurazione della rubrica di *Thunderbird* per l'utilizzo di un server LDAP.

[146] nel caso di *Evolution* ad esempio viene fornito col pacchetto uno schema `evolutionperson.schema` che permette di inserire nell'indirizzario dati aggiuntivi come indirizzi secondari, ecc.

[147] nel caso di *Evolution* poi la cosa è assolutamente da evitare, in quanto di default detto programma, quando lo schema aggiuntivo è presente, salva certi dati negli attributi da esso definiti, anche quando sarebbero disponibili attributi analoghi nella classe standard `inetOrgPerson`.

[148] per un elenco completo occorre sommare quelli della *objectClass* `person` definita in `core.schema`, da cui deriva `organizationalPerson`, definita in `cosine.schema`, da cui finalmente deriva `inetOrgPerson`.

[149] in genere è sempre un qualcosa sotto una *organizationalUnit* a parte, ad esempio `ou=Contacts`.

Per l'utilizzo dei dati della rubrica basta utilizzare un qualunque client LDAP adatto allo scopo, cioè in grado di riconoscere i dati e di presentarli all'utente in maniera opportuna. Alcuni programmi ad interfaccia grafica che sono in grado di interfacciarsi con LDAP per accedere ai dati di una rubrica condivisa sono *Evolution*, *Thunderbird*, *Kaddressbook*.

In tutti questi casi occorre comunque indicare al programma come collegarsi con il server LDAP; ed in genere questo deve essere fatto in maniera esplicita. Tutti i programmi citati hanno una opportuna schermata di configurazione che consente di immettere i dati necessari; le informazioni essenziali sono le coordinate del server e la base della ricerca,[150] tutti i programmi citati consentono inoltre di specificare un eventuale utente (tramite il suo DN) con cui collegarsi al server (richiedendo la password al momento dell'accesso), e di abilitare o meno le connessioni cifrate con SSL.[151] Un esempio di questo tipo di schermata è illustrato in fig. 1.9, dove è riportata quella utilizzata da *Thunderbird*.

1.6.2 Apache e LDAP

L'uso classico dei dati di LDAP da parte di Apache[152] è quello della gestione delle restrizioni di accesso previa autenticazione degli utenti. Invece di confrontare le credenziali di accesso con il classico file generato con `htpasswd`, queste possono essere mantenute su un database LDAP, permettendo così una gestione integrata con gli utenti presenti sul server.

Per poter utilizzare Apache in questo modo è necessario che siano stati caricati gli opportuni moduli; fino ad Apache 2.0 questi erano `mod_ldap` e `mod_auth_ldap`, con Apache 2.2 l'infrastruttura dell'autenticazione è cambiata, e al posto di `mod_auth_ldap` occorre usare `mod_authnz_ldap`. In entrambi i casi comunque questi fanno parte dei moduli base di Apache (in Debian sono nel pacchetto `apache2-common`) e per caricarli basta assicurarsi della presenza degli opportuni link simbolici in `mods-enabled`, creandoli, qualora necessario, con il comando `a2enmod`.

Al solito l'autenticazione si attiva con le direttive `AuthType` e `AuthName`,[153] ma la presenza dei moduli citati consente di rimpiazzare le direttive che specificano l'elenco degli utenti con quelle che permettono utilizzare un server LDAP, e di estendere le capacità della direttiva `Require` così che essa possa usare dati peculiari di LDAP; a questa si accompagnano poi una serie di direttive aggiuntive per gestire le modalità con cui si eseguono le interrogazioni al server.

Il funzionamento dei moduli di autenticazione è infatti suddiviso in due fasi, la prima fase è quella detta di *autenticazione*, che prevede la verifica delle credenziali fornite dall'utente[154] su LDAP. Questo viene fatto eseguendo prima una ricerca sul database con l'interrogazione specificata dalla URL passata come argomento della direttiva `AuthLDAPURL`. La ricerca viene eseguita di default sull'attributo `uid` a partire dalla base specificata nella URL, ma si può scegliere un altro attributo (ad esempio `cn`) o l'applicazione di un filtro di ricerca utilizzando la sintassi estesa delle URL di LDAP illustrata in sez. 1.1.4.

Se la ricerca restituisce un unico risultato, il DN così ottenuto viene usato per collegarsi al server con la password fornita dell'utente: un risultato positivo completa la fase di autenticazione. Si tenga presente che se la ricerca non restituisce esattamente *un* risultato l'accesso viene negato immediatamente: questo significa anche che se non è possibile eseguire una ricerca anonima occorrerà specificare, con le direttive illustrate in tab. 1.35, un DN e relativa password con cui

[150] che è opportuno limitare alla sezione di albero su cui sono mantenuti i dati dell'indirizzario.

[151] in questo caso si tenga presente che deve essere già stato configurato `ldap.conf` come in sez. 1.3.2 per indicare quali certificati usare, dato che i programmi citati non consentono di fare questa impostazione nella loro schermata di configurazione.

[152] esiste anche un modulo, `mod_vhost_ldap` (si veda `http://modvhostldap.alioth.debian.org/`) per mantenere su LDAP le informazioni relative ai *Virtual Host*.

[153] per una trattazione dell'argomento si può consultare sez. 1.4.2 di [WebServ].

[154] vale a dire username e password immesse nel browser.

effettuare il collegamento iniziale per la ricerca, altrimenti la fase di autenticazione non verrà mai completata.

Alla fase di autenticazione segue quella di *autorizzazione*, in cui l'accesso viene consentito o meno a seconda di quanto specificato nella direttiva `Require`. In questo caso però i parametri che si possono indicare alla direttiva sono cambiati nel passaggio da Apache 2.0 ad Apache 2.2, ed i due casi vanno trattati separatamente.

Nel caso di Apache 2.0 la direttiva `Require` continua a supportare come primo argomento i valori classici `valid-user`, che indica un utente generico, `user` che indica uno specifico utente e `group` che indica un gruppo. A questi si aggiungono due nuovi valori, `ldap-attribute` che permette di autorizzare l'accesso sulla base del valore di un attributo dell'utente e `dn` che consente l'accesso specificando direttamente un *Distinguished Name*.

Nel caso di Apache 2.2 invece il sistema di autenticazione richiede l'indicazione esplicita, tramite la direttiva `AuthBasicProvider`, di quale supporto di autorizzazione si intende utilizzare, che nel caso specifico dovrà usare l'argomento `ldap`.[155] Inoltre cambiano gli argomenti da passare alla direttiva `Require`; in particolare si dovranno usare `ldap-user` e `ldap-group` al posto di `user` e `group`.

Quando si effettua l'autorizzazione per utenti e gruppi (quale che sia la versione di Apache usata) essendo i relativi dati mantenuti su LDAP se ne dovranno tenere in considerazione le caratteristiche, che si riflettono sulla forma dei successivi argomenti di `Require`. Come abbiamo accennato l'username che identifica un utente sul browser può assumere le forme più diverse a seconda dell'attributo che si usa nella interrogazione specificata da `AuthLDAPURL`. Pertanto quando si deve indicare un utente occorrerà usare il valore con cui si è fatto la ricerca nella fase di autenticazione; se ad esempio si fosse utilizzata una ricerca sul *Common Name*, le direttive di autorizzazione sarebbero dovute essere qualcosa del tipo di:[156]

```
Require user "Simone Piccardi"
Require user "Christian Surchi"
```

dove si sono usate le virgolette dato che lo spazio viene utilizzato come separatore per indicare gli utenti all'interno di una lista.

Lo stesso tipo di problematica si presenta se si esegue una autorizzazione in base all'appartenenza ad un gruppo: in primo luogo occorrerà identificare quale gruppo indicando come argomento di `require group` (o `require ldap-group`) il *Distinguished Name* della voce ad esso corrispondente; inoltre, dato che per definire dei gruppi si possono usare diversi tipi di *objectclass*, occorrerà anche prendere in considerazione le modalità con cui vengono mantenuti gli utenti all'interno del gruppo.

In questo caso gli aspetti di cui tenere conto sono due. Il primo riguarda quale attributo viene usato per identificare gli utenti; il default usato da `mod_auth_ldap` e `mod_authnz_ldap` è cercare degli attributi di tipo `member` o `uniqueMember`. Dato che questi attributi sono tipici di alcune *objectclass* se ne possono anche specificare di diversi, passando i rispettivi nomi come argomenti della direttiva `AuthLDAPGroupAttribute`. Se ad esempio per i gruppi viene usata una *objectclass* di tipo `posixGroup` sarà necessario utilizzare una direttiva del tipo:

```
AuthLDAPGroupAttribute memberUid
```

dato che in questo caso i membri del gruppo saranno indicati da altrettanti valori di detto attributo.

Il secondo aspetto è quello del valore che viene preso per identificare l'utente nel gruppo; il default di `mod_auth_ldap` (e `mod_authnz_ldap`) prevede infatti che i membri di un gruppo siano

[155] normalmente questa non viene usata dato che il valore di default è `file`.

[156] si è usata la sintassi di Apache 2.0, nel caso di Apache 2.2 si sarebbe dovuto usare `ldap-user`, con le stesse avvertenze riguardo gli argomenti successivi.

indicati con il loro *Distinguished Name*,[157] il controllo di appartenenza pertanto viene effettuato normalmente usando il DN corrispondente all'username fornito dal browser, come lo si è ottenuto dalla ricerca della fase di autenticazione. Di nuovo esistono casi, come quando si usa l'*objectclass* `posixGroup`, in cui i membri del gruppo sono indicati con un semplice nome, in tal caso si dovrà cambiare la modalità di ricerca dei membri con la direttiva `AuthLDAPGroupAttributeIsDN`, impostando:

```
AuthLDAPGroupAttributeIsDN off
```

ed in questo caso verrà usato direttamente l'username ottenuto dal browser.

Nel caso di Apache 2.0 oltre alla autorizzazione per utenti e gruppi, ci sono altre due modalità aggiuntive realizzate da `mod_auth_ldap`; con "`Require dn`" si potrà specificare direttamente (senza usare le virgolette) il *Distinguished Name* degli utenti che si vogliono far accedere, mentre con "`Require attribute`" si potrà specificare un attributo, nella forma `nome=valore`, ed una volta passata la fase di autenticazione, l'autorizzazione sarà concessa soltanto se nella voce corrispondente all'utente è presente quell'attributo, con quel valore.

Qualora si utilizzi Apache 2.2 queste due modalità aggiuntive permangono, con la sola differenza che occorre usare `ldap-dn` al posto di `dn`. Viene però introdotta una ulteriore modalità di autorizzazione grazie all'argomento `ldap-filter`, che permette di effettuare la selezione specificando direttamente un filtro di ricerca, e consente l'accesso a tutti gli utenti autenticati che corrispondono allo stesso. In tab.

Infine occorre tenere presente che `mod_authnz_ldap` non supporta direttamente l'uso dell'argomento `valid-user`. É possibile però utilizzarlo ugualmente, riportandosi nella stessa situazione che si avrebbe con Apache 2.0, impostando la direttiva `AuthzLDAPAuthoritative` ad `off`. In questo caso infatti quando la procedura di autorizzazione su LDAP fallisce, si passa ad utilizzare quella di eventuali altri moduli. Nel caso specifico quello che succede è che la richiesta di autorizzazione viene passata alla procedura di autorizzazione standard tramite file, che supporta `valid-user` e che autorizza un qualunque utente autenticato, e questo vale anche se la procedura di autenticazione era stata fatta su LDAP.

Direttiva	**Significato**
`AuthLDAPURL`	specifica una URL (con la sintassi estesa illustrata in sez. 1.1.4) sulla quale viene compiuta la ricerca degli utenti.
`AuthLDAPBindDN`	specifica il DN con cui collegarsi al server LDAP, se per la ricerca eseguita nella fase di autenticazione non è sufficiente il collegamento anonimo.
`AuthLDAPBindPassword`	specifica la password da usare per eseguire il collegamento con l'utente indicato da `AuthLDAPBindDN`.
`AuthLDAPGroupAttribute`	specifica quali attributi usare per identificare i membri di un gruppo; il default è l'uso di `member` e `uniqueMember`.
`AuthLDAPGroupAttributeIsDN`	esegue la ricerca dei membri di un gruppo usando il DN dell'utente specificato, se disabilitato (con argomento `off`) esegue la ricerca con il nome passato in fase di autenticazione (il default è `on`.
`AuthzLDAPAuthoritative`	direttiva specifica di Apache 2.2; se posta ad `off` consente ad altri moduli di autenticazione di effettuare l'autorizzazione dell'utente quando non funziona quella su LDAP, il default è `on` e deve essere modificato quando si vuole utilizzare la direttiva `Require` con argomento `valid-user`.

Tabella 1.35: Le direttive specifiche della configurazione di Apache per l'autenticazione su LDAP.

In tab. 1.35 si sono riportate le principali direttive di Apache relative all'autenticazione degli utenti su LDAP, la tabella è divisa in due sezioni, nella parte superiore sono indicate le direttive attinenti la fase di autenticazione, nella parte inferiore quelle attinenti la fase di

[157] è il caso tipico della *objectclass* standard `groupOfNames`.

autorizzazione. Per ulteriori dettagli si può consultare la documentazione dei moduli citati, disponibile su `http://httpd.apache.org/docs/`.

1.6.3 Postfix e LDAP

Una delle caratteristiche più interessanti di Postfix è la sua capacità di utilizzare per gran parte dei suoi parametri di configurazione delle tabelle dalle quali estrarre dei valori con una ricerca su un campo che fa da chiave. La potenza di questa caratteristica è che queste tabelle possono essere mantenute sui più vari tipi di supporti: file, hash binari, database relazionali, ed ovviamente anche su LDAP.

Se si installa Postfix con il supporto per LDAP, (su Debian il pacchetto che lo fornisce è `postfix-ldap`) diventa allora possibile inserire le informazioni di una qualunque tabella all'interno di un database, ed ottenerne i valori con delle interrogazioni allo stesso. In questo caso la sintassi[158] con cui fare riferimento ad una tale tabella è analoga alla seguente:

```
alias_maps = ldap:/etc/postfix/ldap-aliases.cf
```

che è del tutto analoga alla sintassi usata per le altre tabelle, ma con l'indicativo "`ldap:`" al posto del più comune "`hash:`" usato per le tabelle ordinarie su DBM. In questo modo è possibile specificare un file (nel precedente esempio `/etc/postfix/ldap-aliases.cf`) da cui vengono prese le impostazioni di accesso a LDAP, e vengono indicati i criteri di ricerca e quali informazioni richiedere.

Questo file supporta la stessa sintassi generale dei file di configurazione di Postfix, che prevede una serie di direttive nella forma `chiave = valore`. Le direttive fondamentali da inserire in questo file sono quattro: la prima è `server_host` che consente di specificare il server LDAP a cui rivolgersi (il default è `localhost`) e supporta anche la sintassi delle URL estese vista in sez. 1.1.4.

La seconda è `search_base` che indica la base della ricerca (secondo quanto visto in sez. 1.2.5) nell'albero, questa può coincidere con la radice del database, ma in genere è opportuno utilizzarla per indicare la sezione dell'albero su cui stanno i dati di interesse.[159]

Espressioni	Significato
`%s`	chiave di ricerca o valore del risultato.
`%u`	quando la chiave di ricerca o il valore del risultato sono nella forma canonica di un indirizzo di posta (`utente@dominio`) restituisce la parte locale dell'indirizzo (nel caso `utente`); altrimenti riporta l'intera chiave.
`%d`	quando la chiave di ricerca o il valore del risultato sono nella forma canonica di un indirizzo di posta (`utente@dominio`) restituisce la parte di dominio dell'indirizzo (nel caso `dominio`); altrimenti non riporta nulla.
`%%`	il carattere `%`.
`%[SUD]`	gli equivalenti maiuscoli di `%s`, `%u` e `%d`.
`%[1-0]`	nel caso di chiave o valore contenente un nome a dominio riporta le componenti dei vari livelli (ad esempio nel caso di `piccardi@truelite.it` `%1` è `it` e `%2` è `truelite`).

Tabella 1.36: La sintassi delle espressioni di sostituzione nelle direttive di LDAP.

[158] questa è la sintassi introdotta con le versioni di Postfix successive alla 2.0, in precedenza usava una notazione diversa, che, come incentivo all'aggiornamento, non tratteremo affatto (essa viene comunque riconosciuta, per compatibilità, nelle nuove versioni).

[159] a partire da Postfix 2.2 si possono usare nel valore di questa direttiva le espansioni di tab. 1.36.

La terza direttiva è `query_filter`, che imposta un filtro di ricerca con il quale si selezionano i risultati che si vogliono estrarre dal database LDAP. Questa direttiva prende come valore un filtro di ricerca (nella forma illustrata in sez. 1.2.5) che supporta una sintassi estesa che permette di utilizzare delle espressioni che fanno riferimento a dati forniti direttamente da Postfix; ad esempio con `%s` si indica la chiave di ricerca passata direttamente dal server (in genere l'indirizzo di posta). Una lista delle espressioni supportate[160] è riportata in tab. 1.36.

La quarta direttiva è `result_attribute`, che permette di indicare quale attributo delle voci ottenute con la ricerca deve essere come valore della tabella; a questa si può aggiungere `result_format` che consente di riformattare il risultato precedente utilizzando le stesse espressioni di tab. 1.36.

Con queste quattro direttive diventa possibile fare eseguire a Postfix una ricerca su LDAP per qualunque direttiva che prenda fra i suoi argomenti una tabella, ad esempio se si vogliono mantenere gli alias degli su LDAP, utilizzando l'attributo `mail` della *objectClass* `inetOrgPerson`, si potrà utilizzare la direttiva usata prima come esempio, ed il contenuto del file `ldap-aliases.cf` sarà qualcosa del tipo:

```
server_host  = 127.0.0.1
search_base  = ou=people,dc=truelite,dc=it
query_filter = mail=%u@truelite.it
result_attribute = uid
version = 3
```

Le altre principali direttive utilizzabili nel file di configurazione di una tabella LDAP sono illustrate in tab. 1.37; un elenco completo delle direttive, insieme ad una descrizione dettagliata del formato del file, sono riportati nella apposita pagina di manuale accessibile con `man ldap_table`.

Direttiva	Significato
`server_host`	indirizzo (o URL) del server LDAP.
`server_port`	porta del server LDAP.
`search_base`	base delle ricerche.
`query_filter`	filtro di ricerca, supporta le estensioni di tab. 1.36.
`result_format`	una stringa di formattazione per i risultati della ricerca.[161]
`result_attribute`	attributo della voce restituita nella ricerca, che viene preso come valore della tabella.
`bind`	indica che occorre collegarsi al server LDAP come utente, prende i valori `yes` o `no`.
`bind_dn`	il *Distinguished Name* dell'utente con cui collegarsi al database.
`bind_pw`	la password dell'utente indicato da `bind_dn`.

Tabella 1.37: Le direttive di configurazione di una tabella di Postfix su LDAP.

1.6.4 Samba e LDAP

Una delle applicazioni più interessanti di LDAP è quella della sua integrazione con Samba, grazie alla quale diventa possibile fornire un servizio di autenticazione centralizzato che può essere utilizzato in contemporanea sia da sistemi Unix che da sistemi Windows. È infatti possibile indicare a Samba di inserire le informazioni con cui effettua la gestione di un *Dominio* Windows, comprese quelle dei relativi utenti, su LDAP.

[160] la lista fa riferimento alla versione 2.2 di Postfix; nelle versioni precedenti erano disponibili solo le prime tre espressioni.

[161] per le versioni precedenti Postfix 2.2 il nome di questa direttiva era `result_filter`, ma poi è stato cambiato data la sua ambiguità.

Si tenga presente però che l'autenticazione centralizzata nel mondo Windows è fondata su concetti completamente diversi rispetto a quanto utilizzato in ambito Unix, e questo porta ad una lunga serie di complicazioni. Una prima complicazione è che nel mondo Windows gli utenti di un *Domain Controller*[162] sono di due tipi, gli utenti associati ad una persona, del tutto analoghi agli utenti Unix, e quelli associati ad una macchina. Samba deve poter mappare entrambi i tipi utenti di un dominio Windows su altrettanti utenti ordinari in una macchina Unix. Questo ad esempio comporta, nell'inserire i relativi dati su LDAP, la necessità di avere due rami distinti, uno per gli utenti ordinari, l'altro per le macchine.

Una seconda complicazione è quella relativa alla gestione degli identificativi, che su Windows hanno una forma completamente diversa, in cui ogni "*oggetto*" (utente, macchina, gruppo) è associato ad un diverso identificativo (il cosiddetto *Security ID* o SID) che si suppone univoco a livello globale.[163] Questo può comportare problemi nella migrazione dei dati da un server ad un altro, dato che in genere fra due istanze diverse di Samba i SID non corrispondono.

La direttiva principale che indica a Samba di registrare le sue informazioni su LDAP[164] è `passdb backend`, che nel caso deve indicare l'utilizzo di LDAP attraverso la parola chiave `ldapsam`, seguita dalle modalità con cui contattare il server. Un esempio di questa direttiva è il seguente:

```
passdb backend = ldapsam:ldap://127.0.0.1/
```

Per poter eseguire la manutenzione dei suoi dati Samba necessita di poter accedere al server LDAP con un utente che abbia adeguati privilegi, che si indica con la direttiva `ldap admin dn`; si dovrà inoltre specificare dove i dati vengono mantenuti all'interno dell'albero. Per la configurazione di tutte queste proprietà esistono una serie di direttive, introdotte tutte dalla parola chiave `ldap`, la principali delle quali sono illustrate in tab. 1.38; per un elenco completo si faccia riferimento alla pagina di manuale, accessibile con `man smb.conf`.

Un esempio delle direttive necessario, per quanto riguarda la riguarda la configurazione dell'utilizzo di LDAP, è il seguente estratto di `smb.conf`, preso dalla versione installata dal citato pacchetto `truelite-fileserver`:

```
passdb backend = ldapsam:ldap://127.0.0.1/
ldap passwd sync = yes
ldap admin dn = cn=admin,dc=truelite,dc=it
ldap suffix = dc=truelite,dc=it
ldap group suffix = ou=Groups
ldap user suffix = ou=Users
ldap machine suffix = ou=Computers
```

Si noti che fra le direttive di tab. 1.38 non ne esiste nessuna che indichi la password con la quale collegarsi al server LDAP.[166] Questa infatti deve essere impostata manualmente con il comando `smbpasswd`, che prevede l'uso dell'apposita opzione `-w` che memorizza nei file di gestione di Samba[167] la password da usare in corrispondenza al DN indicato dalla direttiva `ldap admin dn` per collegarsi al server LDAP. Per far si che Samba sia in grado di appoggiarsi a LDAP si dovrà dunque eseguire a parte il comando:

```
smbpasswd -w password_difficile_e_segreta
```

[162] non entreremo nei dettagli della terminologia Windows, che si suppone nota.

[163] la parte iniziale del SID è diversa fra dominio e dominio, quindi la stessa macchina o lo stesso utente avranno identificativi diversi in domini diversi.

[164] tutte le direttive di Samba relative all'uso di LDAP devono essere inserite nella sezione `global` di `smb.conf`.

[165] quella degli `idmap` è una funzionalità di Samba che permette di memorizzare le corrispondenze fra gli indicativi usati da Windows, e gli ID usati su sistemi Unix.

[166] non esiste cioè l'analogo di `bindpw` che avevamo visto in sez. 1.5.2.

[167] per la precisione in `/var/lib/secrets.tdb`.

Direttiva	Significato
`ldap suffix`	il suffisso che il ramo di albero contenente i dati che Samba mantiene su LDAP; prende come argomento il relativo DN.
`ldap user suffix`	la base della ricerca dei dati relativi agli utenti; specificato come RDN rispetto al suffisso generale dato da `ldap suffix` (in genere è qualcosa del tipo `ou=users`).
`ldap group suffix`	la base della ricerca dei dati relativi ai gruppi di utenti, con la stessa sintassi del precedente (è in genere qualcosa del tipo `ou=group`).
`ldap machine suffix`	la base della ricerca dei dati relativi agli account associati alle macchine del dominio (è in genere qualcosa del tipo `ou=computers`).
`ldap idmap suffix`	la base della ricerca dei dati relativi alle alla mappatura degli identificatori.[165]
`ldap admin dn`	il *Distinguished Name* dell'utente usato da Samba per operare sul server LDAP.
`ldap delete dn`	specifica se una operazione di cancellazione di un utente cancella tutta la voce su LDAP o soltanto la parte relativa a Samba.
`ldap passwd sync`	specifica se eseguire la sincronizzazione della password Unix quando viene modificata quella Windows.
`ldap ssl`	indica se per il collegamento al server deve essere usato SSL.

Tabella 1.38: Le direttive di Samba per la configurazione su LDAP.

Ovviamente oltre a dire a Samba di utilizzare LDAP, occorrerà che sul server siano disponibili le opportune *objectClass* che definiscono gli attributi in cui mantenere le informazioni usate da Samba. Per questo insieme a Samba viene distribuito il file `samba.schema`;[168] detto file dovrà essere copiato in `/etc/ldap/schema` e si dovrà riconfigurare `slapd.conf` aggiungendo alle altre direttive di inclusione degli schemi una riga del tipo:

```
include       /etc/ldap/schema/samba.schema
```

In tab. 1.39 si sono illustrati i principali attributi della *objectclass* `sambaSamAccount`,[169] che è l'analoga della `posixAccount` vista in sez. 1.5.1 per gli utenti Unix.

Un secondo requisito è che l'utente specificato dalla direttiva `ldap admin dn` abbia sufficienti privilegi di accesso; per questo, e per permettere agli utenti di modificare le proprie informazioni, dovranno essere impostate delle opportune ACL; un estratto di quanto è necessario aggiungere a `slapd.conf` è il seguente:[171]

```
access to attrs=userPassword,sambaNTPassword,sambaLMPassword,sambaPwdLastSet,
  sambaPwdMustChange,sambaPasswordHistory,shadowLastChange
        by dn="cn=admin,dc=truelite,dc=it" write
        by anonymous auth
        by self write
        by * none
```

Infine, come per la centralizzazione degli utenti Unix, anche in questo caso è necessario che le informazioni siano adeguatamente strutturate all'interno dell'albero. Questa struttura è analoga

[168] nel caso di Debian è distribuito col pacchetto `samba-doc`, e viene installato in forma compressa sotto `/usr/share/samba-doc/examples/LDAP/samba.schema.gz`

[169] in realtà vengono definite varie *objectclass* per poter mantenere su LDAP anche altri dati relativi alla gestione di un dominio.

[170] tutti i tempi usati da Samba in questo e negli altri attributi sono espressi nel cosiddetto *unix-time*, vale a dire in numero di secondi a partire dal 1 Gennaio 1970.

[171] occorrerà anche, se si vogliono avere prestazioni accettabili, indicizzare opportunamente gli attributi coinvolti; per questo si lascia comunque spazio alla documentazione reperibile in numerosi HOWTO, o alla versione di `slapd.conf` usata nel pacchetto `truelite-fileserver` disponibile su `http://debian.truelite.it`.

Attributo	Significato
`sambaNTPassword`	password nel formato NT (hash MD4 della password in Unicode).
`sambaLMPassword`	password nel formato LAN manager.
`sambaPwdLastSet`	tempo di ultima modifica delle password.[170]
`sambaPwdMustChange`	tempo in cui la password dovrà essere modificata.
`sambaKickoffTime`	tempo in cui l'account sarà bloccato, analogo dell'attributo `shadowExpire` di `shadowAccount` (vedi tab. 1.30).
`sambaPasswordHistory`	password precedenti, non potranno essere riutilizzate al momento della modifica della password.
`sambaSID`	*Security ID* (SID) dell'utente.
`sambaPrimaryGroupSID`	*Security ID* (SID) del gruppo principale dell'utente.
`sambaAcctFlags`	flag, una stringa di 11 caratteri delimitata da parentesi quadre che definisce le caratteristiche dell'account.
`sambaDomainName`	dominio Windows di cui l'utente fa parte.

Tabella 1.39: I principali attributi della *objectclass* `sambaSamAccount` usata da Samba per mantenere i dati di un utente.

a quella già vista in sez. 1.5.1, che può anche essere riutilizzata per la parte di utenti e gruppi;[172] ma come accennato si avrà necessità di almeno di un ramo ulteriore per mantenere gli elenchi delle macchine nel dominio.[173]

Si tenga presente che molti dei dati utilizzati da Samba sono del tutto indipendenti rispetto agli analoghi di sez. 1.5 per le credenziali degli utenti Unix, in particolare Samba utilizza dei dati propri sia per gli identificativi che per le password. Questo comporta una ulteriore complicazione, se infatti lo scopo è quello di unificare le autenticazioni in un ambiente misto, occorrerà assicurarsi che le credenziali degli utenti (ed in particolare la password) siano sempre le stesse, sia per la parte Windows (gestita da Samba) che per quella Unix (gestita da PAM).

Il problema che può sorgere è che un utente Unix cambi la sua password, ad esempio con `passwd`, modificando su LDAP solo l'attributo `userPassword`, mentre la relativa password per Windows (che viene mantenuta in altri attributi) resta invariata. Nel caso opposto è possibile dire a Samba, con la direttiva `ldap passwd sync`, di mantenere sincronizzate tutte le password, ma questo comporta comunque che in un ambiente misto operazioni come il cambiamento della password debbano essere gestite in maniera opportuna.

Per questo motivo esistono vari programmi di ausilio che consentono di eseguire una gestione *unificata* degli utenti, tenendo conto automaticamente di queste problematiche di sincronizzazione. In questo ambito i più utilizzati sono i cosiddetti `smbldap-tools`,[174] un insieme di script in Perl che consente di gestire su LDAP le credenziali di autenticazione per un ambiente misto Unix/Windows.[175]

Questi script ricalcano il principio di funzionamento del programma `cpu` illustrato in sez. 1.5.1, e forniscono una serie di comandi (`smbldap-useradd`, `smbldap-userdel`, `smbldap-usermod` e gli analoghi per i gruppi) che hanno una sintassi praticamente identica ai relativi comandi classici, ma effettuano la gestione di utenti e gruppi sull'albero LDAP, tenendo anche conto (a differenza di quanto avviene per `cpu`) della necessità di mantenere anche le informazioni usate da Samba.

Dato che questi programmi operano direttamente sui dati di LDAP, occorre configurarli opportunamente in modo da consentire loro l'accesso; questo viene fatto attraverso due file di configurazione in `/etc/smbldap/`. Il primo file è `smbldap_bind.conf`, che contiene esclusiva-

[172] è quello che si fa normalmente quando si vuole usare LDAP per l'autenticazione centralizzata sia su Unix che su Windows.

[173] e di un altro ramo qualora si intenda usare gli *idmap*.

[174] in Debian sono installabili direttamente tramite l'omonimo pacchetto.

[175] con la versione 3.0.23 di Samba alcune di queste funzionalità sono state inserite direttamente all'interno di Samba, e possono essere gestite con il comando `net`; essendo questa funzionalità estremamente recente (al momento della stesura di questa sezione, Agosto 2006) non le prenderemo in esame.

mente i dati da usare per il collegamento al server, vale a dire il DN dell'utente con cui collegarsi e la relativa password. Dato che i comandi supportano l'utilizzo di due server (un master ed uno slave) esistono in tutto quattro direttive da usare, illustrate in tab. 1.40.

Direttiva	Significato
`masterDN`	il *Distinguished Name* dell'utente usato per collegarsi al server LDAP.
`slaveDN`	il *Distinguished Name* dell'utente usato per collegarsi ad un eventuale server LDAP secondario.
`masterPw`	la password dell'utente usato per collegarsi al server LDAP principale.
`slavePw`	la password dell'utente usato per collegarsi ad un eventuale server LDAP secondario.

Tabella 1.40: Le direttive di configurazione del file `smbldap_bind.conf`.

Siccome i dati mantenuti in `smbldap_bind.conf` prevedono la presenza della password di accesso in chiaro, detto file deve essere protetto da lettura. Si tenga conto che l'esistenza di un server secondario è del tutto opzionale, qualora non esista basterà usare gli stessi valori indicati per il primario; un esempio del contenuto di questo file è il seguente:

```
slaveDN="cn=admin,dc=truelite,dc=it"
slavePw="password_segretissima"
masterDN="cn=admin,dc=truelite,dc=it"
masterPw="altra_password_altrettanto_segreta"
```

Tutte le restanti configurazioni sono invece mantenute nel file `smbldap.conf`, a partire dall'indicazione del server a cui rivolgersi, del ramo di albero su cui sono mantenute le informazioni, le modalità per eseguire le ricerche ed una serie di valori di default da utilizzare nella gestione di utenti e gruppi e per la configurazione di varie caratteristiche di Samba.

Il file `smbldap.conf` è sostanzialmente uno spezzone di codice Perl che viene incluso dai vari comandi dei `smbldap-tools`, per cui le *direttive* sono tutte nella forma di una assegnazione di variabile (giacché in realtà proprio di questo si tratta) in cui il valore assegnato è fra virgolette.[176]

Le principali direttive (o variabili, a seconda del punto di vista) sono illustrate brevemente in tab. 1.41, il file di configurazione installato di default normalmente è ampiamente commentato con una descrizione dettagliata di ciascuna variabile.[177]

Le direttive essenziali sono `masterLDAP` e `masterPort` che permettono di indicare l'indirizzo del server (e rispettiva porta) a cui rivolgersi, la maggior parte delle altre (quelle riportate nella prima parte di tab. 1.41) servono ad impostare le caratteristiche della connessione al server, (ad esempio `suffix` indica la base dell'albero, e le varie `*dn` specificano poi i sottorami in cui sono mantenute le varie informazioni). Una serie di altre direttive (quelle riportate nella parte finale di tab. 1.41) permettono invece di impostare caratteristiche standard degli utenti creati con i comandi degli `smbldap-tools`, (ad esempio la shell di login con `userLoginShell`, o la directory da usare per mantenere i profili Windows con `userProfile`).

Un ruolo particolare è però quello svolto dalla direttiva `SID`, essa infatti indica il *Security ID* del dominio Samba; e serve come suffisso a tutti gli identificativi usati in ambito Windows. Questo valore non può essere assegnato a piacere, ma deve corrispondere a quello usato da Samba, pertanto prima di poter usare gli `smbldap-tools` occorrerà eseguire il comando:

```
net getlocalsid
```

[176] e nell'assegnazione si può anche fare riferimento al valore di un'altra variabile con la sintassi `${variabile}`.
[177] almeno questo è quello che avviene per la versione installata da Debian.

Direttiva	Significato
masterLDAP	indirizzo IP del server LDAP.
masterPort	porta del server LDAP.
slaveLDAP	indirizzo IP di un eventuale server LDAP secondario.
slavePort	porta di un eventuale server LDAP secondario.
suffix	la base della sezione di albero su cui sono mantenuti i dati degli account; in genere coincide con la base dell'albero stesso.
usersdn	il DN della sezione di albero su cui sono mantenuti i dati degli utenti.
groupsdn	il DN della sezione di albero su cui sono mantenuti i dati dei gruppi.
computersdn	il DN della sezione di albero su cui sono mantenuti i dati delle macchine.
scope	la *profondità* della ricerca (secondo i valori di tab. 1.17).
SID	il suffisso dei *Security ID* del dominio.
userLoginShell	la shell di default assegnata ai nuovi utenti.
defaultUserGid	il *group-id* del gruppo di default assegnato ai nuovi utenti.
defaultComputerGid	il *group-id* del gruppo di default assegnato agli utenti usati per identificare le macchine nel dominio.
skeletonDir	la directory da cui prendere i file da mettere nella home di un nuovo utente.
userSmbHome	lo share su cui viene mappata la directory home dell'utente.
userProfile	lo share su cui sono mantenuti i profili degli utenti.
userHomeDrive	il disco Windows su cui viene mappato la directory home dell'utente.
userScript	lo script di avvio per Windows (preso dalla home dell'utente) che viene eseguito all'ingresso nel dominio.

Tabella 1.41: Le direttive di configurazione del file `smbldap.conf`.

sulla macchina su cui gira il server Samba, per ottenere il valore da assegnare a questa variabile.[178] Questo infatti deve corrispondere ai valori che Samba mantiene internamente, altrimenti i dati di LDAP saranno considerati come appartenenti ad utenti di un altro dominio ed ignorati.

Se si avesse un albero già popolato di dati (e quindi con una serie di SID già determinati) è comunque possibile forzare Samba ad utilizzare un SID diverso con il comando:

```
net setlocalsid S-1-5-21-...-...-...
```

Per concludere un estratto del contenuto del file `smbldap.conf`, preso da una configurazione creata grazie al pacchetto `truelite-fileserver` citato in precedenza, è il seguente:

```
SID="S-1-5-21-1911238739-97561441-2706018148"
slaveLDAP="127.0.0.1"
slavePort="389"
masterLDAP="127.0.0.1"
masterPort="389"
...
usersdn="ou=Users,${suffix}"
computersdn="ou=Computers,${suffix}"
groupsdn="ou=Groups,${suffix}"
idmapdn="ou=Idmap,${suffix}"
...
userLoginShell="/bin/bash"
userHome="/home/%U"
userGecos="System User"
```

[178] in genere il *Security ID* viene impostato autonomamente da Samba quando lo si configura come server PDC.

```
defaultUserGid="513"
defaultComputerGid="515"
skeletonDir="/etc/skel"
defaultMaxPasswordAge="175"
userSmbHome=""
userProfile=""
userHomeDrive="H:"
userScript="%U.cmd"
...
```

Una volta completata la configurazione in genere il primo passo che è quello di usare il comando `smbldap-populate`, che permette di creare su LDAP l'infrastruttura iniziale necessaria per gestire gli utenti di un dominio Windows, in maniera analoga alla procedura che si è vista in sez. 1.5.1 con l'uso dei `migrationtools`.

Una delle caratteristiche del comando è quella di creare, oltre ad una serie di gruppi standard dei domini Windows, un utente ospite per l'accesso anonimo (il default è `nobody`, ma lo si può cambiare specificando un nome diverso con l'opzione `-b`) ed un utente amministratore del dominio (il default è `Admininistrator`, che come prima può essere cambiato, usando l'opzione `-a`). Quest'ultimo è molto importante, dato che è l'utente per conto del quale si potranno aggiungere macchine al dominio. Altre due opzioni supportate sono `-e` che invece di inserire i dati li esporta su un file LDIF (specificato come parametro dell'opzione) e `-i` che crea l'infrastruttura leggendola da un file LDIF (sempre passato come parametro).

Alla sua esecuzione il comando chiede la password dell'amministratore (l'utente ospite è senza password) ed esegue la creazione dell'infrastruttura notificando a video la creazione delle varie voci, con un qualcosa del tipo:

```
monk:~# smbldap-populate
Using workgroup name from sambaUnixIdPooldn (smbldap.conf): sambaDomainName=DOMINIO
Using builtin directory structure
entry dc=truelite,dc=it already exist.
adding new entry: ou=Users,dc=truelite,dc=it
adding new entry: ou=Groups,dc=truelite,dc=it
adding new entry: ou=Computers,dc=truelite,dc=it
adding new entry: ou=Idmap,dc=truelite,dc=it
entry sambaDomainName=DOMINIO,dc=truelite,dc=it already exist. Updating it...
adding new entry: uid=admin,ou=Users,dc=truelite,dc=it
adding new entry: uid=nobody,ou=Users,dc=truelite,dc=it
adding new entry: cn=Domain Admins,ou=Groups,dc=truelite,dc=it
adding new entry: cn=Domain Users,ou=Groups,dc=truelite,dc=it
adding new entry: cn=Domain Guests,ou=Groups,dc=truelite,dc=it
adding new entry: cn=Domain Computers,ou=Groups,dc=truelite,dc=it
adding new entry: cn=Administrators,ou=Groups,dc=truelite,dc=it
adding new entry: cn=Print Operators,ou=Groups,dc=truelite,dc=it
adding new entry: cn=Backup Operators,ou=Groups,dc=truelite,dc=it
adding new entry: cn=Replicators,ou=Groups,dc=truelite,dc=it
Changing password for admin
New password :
Retype new password :
```

Una volta che l'infrastruttura dell'albero è a posto, i vari comandi del pacchetto consentiranno di creare, cancellare e modificare i dati di utenti e gruppi operando direttamente su LDAP, mantenendo allo stesso tempo la coerenza degli stessi per il loro utilizzo sia da parte di Samba che da parte di PAM e NSS in modo da avere un meccanismo di gestione centralizzata sia in ambiente Unix che Windows; in tab. 1.42 si è riportato un riassunto dei vari comandi che fanno parte degli `smbldap-tools`.

I comandi per creare, modificare e cancellare un utente sono rispettivamente `smbldap-useradd`, `smbldap-usermod` e `smbldap-userdel`. Quest'ultimo prende come unica opzione `-r`, che se specificata rimuove anche la home directory dell'utente. I primi due supportano un

Comando	Scopo
`smbldap-populate`	crea l'infrastruttura iniziale dei dati su LDAP.
`smbldap-useradd`	crea un nuovo utente.
`smbldap-usermod`	modifica le caratteristiche di un utente.
`smbldap-userdel`	cancella un utente.
`smbldap-groupadd`	aggiunge un gruppo.
`smbldap-groupmod`	modifica le caratteristiche di un gruppo.
`smbldap-groupdel`	cancella un gruppo.
`smbldap-passwd`	cambia la password di un utente.

Tabella 1.42: I comandi che fanno parte del pacchetto `smbldap-tools`.

insieme comune di opzioni, le principali delle quali sono illustrate in tab. 1.43. La tabella è suddivisa in due sezioni, in alto si sono riportate le opzioni relative all'ambiente Unix, simili a quelle degli analoghi comandi classici, ed in basso quelle relative all'ambiente Windows. Per un elenco completo si può al solito fare riferimento alle pagine di manuale dei vari comandi.[179]

Opzione	Significato
`-g`	imposta il gruppo principale dell'utente, se non specificato viene usato il valore corrispondente a quanto indicato da `defaultUserGid`.
`-G`	imposta una lista (specificata come elenco separato da virgole) di gruppi ausiliari di cui l'utente è membro, se non specificato l'utente sarà membro solo del suo gruppo principale.
`-P`	termina invocando `smbldap-passwd` per impostare una password per l'utente.
`-s`	imposta la shell passata come parametro come shell di default.
`-u`	imposta un valore numerico (passato come parametro) per l'*uid* dell'utente.
`-a`	aggiunge le *objectClass* e gli attributi necessari per mantenere i valori delle credenziali in ambito Windows.
`-B`	imposta la proprietà che richiede all'utente Windows il cambiamento della password al primo accesso; prende rispettivamente i valori 0 (no) e 1 (si).
`-C`	imposta lo share su cui viene mappata la directory home dell'utente.
`-D`	imposta la lettera associata al disco Windows su cui viene mappata la directory home dell'utente.
`-E`	imposta lo script Windows eseguito all'ingresso nel dominio.
`-F`	imposta lo share usata per mantenere i profili dell'utente.

Tabella 1.43: Le opzioni comuni dei comandi `smbldap-useradd` e `smbldap-usermod`.

Il comando `smbldap-useradd` prende come argomento l'username dell'utente da creare, e rispetto a quelle già illustrate in tab. 1.43 prevede alcune opzioni specifiche, riportate in tab. 1.44. In genere lo si invoca semplicemente con `-m` per richiedere la creazione della home directory e con `-a` per avere un utente utilizzabile anche in ambito Windows.

Opzione	Significato
`-m`	crea la home directory dell'utente se non esiste già.
`-k`	copia i file della directory definita da `skeletonDir` nella home dell'utente, altrimenti viene usata `/etc/skel` (valida solo se congiunta a `-m`).
`-w`	crea un utente speciale da associare ad una workstation nel dominio (ad uso di Samba per la gestione delle macchine nel dominio).

Tabella 1.44: Le opzioni specifiche del comando `smbldap-useradd`.

[179] si tenga presente però che, almeno alla data di stesura di queste dispense (Agosto 2006) la documentazione contenuta nelle suddette pagine è piuttosto carente.

Per modificare le caratteristiche di un utente già presente si può invece usare il comando `smbldap-usermod` che prende come argomento il nome dell'utente. Le principali opzioni specifiche di questo comando sono riportate in tab. 1.45.

Opzione	Significato
`-e`	imposta una data di scadenza dell'account.
`-I`	disabilita l'utente.
`-J`	abilita l'utente.

Tabella 1.45: Le opzioni specifiche del comando `smbldap-usermod`.

Infine per modificare la password di un utente, con la sicurezza che questa venga aggiornata sia per quanto riguarda la parte usata in ambiente Unix e per LDAP stesso (in sostanza l'attributo `userPassword`) che per l'ambiente Windows (gli attributi usati da Samba allo scopo), è disponibile il comando `smbldap-passwd`, che prende come argomento il nome utente e chiede (due volte, la seconda per conferma) la nuova password. Il comando prende solo due opzioni: `-u` per modificare solo la password Unix e `-s` per modificare solo quella Windows.

Come avviene per comandi classici anche negli `smbldap-tools` sono presenti degli opportuni script per operare sui gruppi; in particolare i comandi analoghi ai precedenti visti per gli utenti sono `smbldap-groupadd`, `smbldap-groupmod` e `smbldap-groupdel`; quest'ultimo permette di cancellare un gruppo, passato come argomento, e non ha nessuna opzione, gli altri due invece supportano alcune opzioni comuni, le principali delle quali riportate in tab. 1.46.

Opzione	Significato
`-g`	imposta il valore numerico del *gid* al valore passato come parametro.
`-a`	aggiunge automaticamente al gruppo un SID opportuno per renderlo utilizzabile in ambito Windows.
`-s`	imposta il valore del SID del gruppo al valore passato come parametro.

Tabella 1.46: Le opzioni comuni dei comandi `smbldap-groupadd` e `smbldap-groupmod`.

Un nuovo gruppo può essere creato con il comando `smbldap-groupadd`, specificando il nome del gruppo. In genere l'unica opzione che si usa è `-a` per richiedere al comando anche la creazione dei dati che verranno utilizzati da Samba per gestire le credenziali per Windows. Altre caratteristiche possono essere impostate manualmente con le opzioni di tab. 1.46.

Opzione	Significato
`-x`	elimina dal gruppo la lista degli utenti (separata da virgole) passata come parametro.
`-m`	aggiunge al gruppo la lista degli utenti (separata da virgole) passata come parametro.
`-n`	modifica il nome del gruppo sostituendolo con quello passato come parametro.

Tabella 1.47: Le opzioni specifiche del comando `smbldap-groupmod`.

Una volta creato un gruppo può essere modificato con il comando `smbldap-groupmod`, che di nuovo prende come argomento il nome del gruppo. Oltre a quelle illustrate in tab. 1.46 le altre opzioni rilevanti specifiche di questo comando sono state elencate in tab. 1.46.

1.6.5 Squid e LDAP

Come proxy per l'accesso alle pagine web, una delle caratteristiche di Squid è quella di consentire l'accesso al servizio solo ad utenti autenticati, ed una delle modalità di autenticazione supportate è appunto quella che ricorre ai dati mantenuti su un server LDAP.

Per utilizzare questa funzionalità occorre anzitutto richiedere a Squid di abilitare l'autenticazione degli utenti; per poter usare utenti mantenuti su LDAP l'unico *schema*[180] disponibile è `basic`.[181] La direttiva di base che permette di impostare l'autenticazione degli utenti è `auth_param`, che prende come primo argomento lo schema da utilizzare (nel nostro caso `basic`), e come terzo argomento il parametro da configurare.

La direttiva deve essere ripetuta per ciascun parametro che si intende configurare, ma quello principale è `program`, che consente di indicare il programma da utilizzare per autenticare gli utenti, che nel caso si voglia usare LDAP è appunto `ldap_auth`; un esempio possibile di uso di questa configurazione è allora quello presa dal seguente estratto di `squid.conf`:

```
auth_param basic children 5
auth_param basic credentialsttl 30 minutes
auth_param basic realm Truelite Proxy Server
auth_param basic program /usr/lib/squid/ldap_auth \
    -b "ou=Users,dc=truelite,dc=it" -v3 -f(uid=%s) -h localhost
```

Nel nostro caso, volendo autenticare gli utenti su LDAP, ricorreremo all'apposito programma `ldap_auth` (distribuito insieme con Squid) il cui unico scopo è interrogare il server per autenticare l'utente. Il programma richiede come opzione obbligatoria `-b` che specifica la base dell'albero, e di default si collega su *localhost*. Si può specificare un server diverso per indirizzo con `-h` o con una URL estesa (quelle di tab. 1.1) con `-H`; le altre principali opzioni sono riportate in tab. 1.48.

Opzione	Significato
`-b`	specifica la base dell'albero, passata come parametro.
`-f`	consente di impostare un filtro, passato come parametro, con cui eseguire la ricerca degli utenti.
`-u`	consente di impostare l'attributo, passato come parametro, da utilizzare per la ricerca del nome utente.
`-s`	consente di specificare la profondità della ricerca, prende come parametro uno dei valori di tab. 1.17.
`-D`	specifica un eventuale DN con cui collegarsi al server LDAP.
`-h`	specifica l'indirizzo del server LDAP.
`-H`	imposta la URL a cui contattare il server LDAP.
`-p`	specifica la porta da utilizzare per la connessione.
`-v`	imposta la versione di protocollo da usare (prende i valori 2 e 3, ed il default, da cambiare, è 2).

Tabella 1.48: Le opzioni del programma `ldap_auth` usato da Squid per l'autenticazione su LDAP.

Una volta abilitata l'autenticazione degli utenti, ed istruito Squid ad eseguire la ricerca degli stessi su LDAP, sarà possibile impostare il controllo di accesso per fornire il servizio solo agli utenti autenticati. Per farlo occorre anzitutto impostare una ACL di tipo `proxy_auth`, ed utilizzarla in una direttiva di accesso, cioè utilizzare qualcosa del tipo:[182]

```
acl password proxy_auth REQUIRED
http_access allow password
```

Se tutto quello che interessa è consentire un accesso generico agli utenti autenticati questo è sufficiente, Squid però consente anche un controllo molto più dettagliato, che permette di

[180] Squid supporta tre schemi di autenticazione, `basic`, `ntml` e `digest`, corrispondenti a tre diversi metodi di richiedere le credenziali al client. Si tenga presente che l'autenticazione `basic` trasmette in chiaro le credenziali stesse, per cui si possono avere degli ovvi problemi di sicurezza.

[181] in realtà appoggiandosi a Samba (vedi sez. 1.6.4) si potrebbe usare anche l'autenticazione NTML, ma questo non riguarda più l'interazione fra Squid e LDAP, ma fra Squid ed un generico PDC Windows.

[182] si ricordi anche che le regole che usano una ACL di tipo `proxy_auth` devono usualmente essere specificata per ultime, in quanto una volta determinato che l'utente è valido, una tale regola consente l'accesso e tutto è permesso; occorre quindi impostare in precedenza le eventuali ulteriori restrizioni (ad esempio sui siti non raggiungibili) da applicare agli utenti autenticati.

impostare ACL diverse a seconda del gruppo di cui gli utenti fanno parte, anch'esso mantenuto su LDAP. Per questo occorre sfruttare una caratteristica particolare di Squid che è quella che gli permette di ampliare le ACL disponibili appoggiandosi ad un programma esterno, in maniera analoga a quanto avviene per eseguire l'autenticazione.

Per far questo le ACL di Squid comprendono una classe speciale, `external`, che consente di effettuare un controllo invocando un opportuno programma di ausilio definito attraverso la direttiva, `external_acl_type`. Quest'ultima prende come argomento un nome, che identificherà quel particolare tipo di ACL, seguita dalle indicazioni necessarie ad invocare il programma esterno.

Nel caso si voglia usare come ACL l'appartenenza di un utente ad un gruppo su LDAP si può utilizzare un apposito programma, `squid_ldap_group`, distribuito insieme a Squid,[183] che permette di effettuare questa verifica. Un possibile esempio di configurazione è allora quello del seguente estratto di `squid.conf`:

```
external_acl_type ldap_group %LOGIN /usr/lib/squid/squid_ldap_group \
    -b "ou=Groups,dc=truelite,dc=it" -B "ou=Users,dc=truelite,dc=it" \
    -f "(&(uniqueMember=%u)(cn=%g))" -h localhost
acl normal_access external ldap_group normal
acl restricted_access external ldap_group restricted
acl full_access external ldap_group full
```

dove si definisce un tipo di ACL esterna `ldap_group`, che fa ricorso al suddetto programma, e poi lo si utilizza per creare delle nuove ACL (le varie `*_access` dell'esempio) da usare per imporre diverse restrizioni a seconda dell'appartenenza ad un certo gruppo (nel caso uno fra `normal`, `restricted`, `full`).

Opzione	Significato
`-b`	specifica la sezione dell'albero, passata come parametro, nella quale sono mantenuti i dati dei gruppi.
`-B`	specifica la sezione dell'albero, passata come parametro, nella quale sono mantenuti i dati degli utenti (se diversa da quella dei gruppi).
`-f`	consente di impostare un filtro, passato come parametro, con cui eseguire la ricerca degli utenti appartenenti ad un gruppo; nel filtro si potrà indicare l'utente con `%u` ed il gruppo con `%g`.
`-s`	consente di specificare la profondità della ricerca, prende come parametro uno dei valori di tab. 1.17.
`-D`	specifica un eventuale DN con cui collegarsi al server LDAP qualora non fossero consentite ricerche anonime.
`-w`	specifica la password da usare per collegarsi al server LDAP (con l'utente specificato da `-D`).
`-h`	specifica l'indirizzo del server LDAP.
`-H`	imposta la URL a cui contattare il server LDAP.
`-p`	specifica la porta da utilizzare per la connessione.
`-v`	imposta la versione di protocollo da usare (prende i valori 2 e 3, ed il default è 2).

Tabella 1.49: Le opzioni del programma `squid_ldap_group` per la creazione di ACL sui gruppi.

Il programma `squid_ldap_group` prende in input una riga contenente utente e gruppo e restituisce il risultato (come richiesto per l'uso da parte di `external_acl_type`) sullo standard output,[184] per cui può essere anche usato da riga comando per verificarne il funzionamento. Come per il precedente `ldap_auth` il comando supporta diverse opzioni, le principali delle quali sono illustrate in tab. 1.49.

La complessità dell'uso di `squid_ldap_group` sta nel modo in cui viene fatta la ricerca della presenza di un utente in un gruppo, che è governata dal filtro di ricerca specificato con l'opzione

[183] almeno questo è quanto avviene per il pacchetto Debian.

[184] la risposta può essere `OK` se l'utente corrisponde, o `ERR` se non corrisponde.

`-f`. Il filtro infatti supporta una sintassi speciale nella quale si possono usare le notazioni `%u` e `%g` per indicare rispettivamente il nome dell'utente ed il nome del gruppo ottenuti in ingresso. Qualora si trovi una voce corrispondente il programma avrà successo.

Appendice A

GNU Free Documentation License

Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be

a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

A.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

A.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

A.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License or any later version applies to it, you have the option of following the terms and conditions either of that specified version or of any

later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Indice analitico

Apache
  direttiva
    AuthLDAPBindDN, 66
    AuthLDAPBindPassword, 66
    AuthLDAPCompareDNOnServer, 66
    AuthLDAPGroupAttributeIsDN, 66
    AuthLDAPGroupAttribute, 66
    AuthLDAPURL, 66
    AuthzLDAPAuthoritative, 66

comando
  cpu, 55
  ldap_auth, 77
  ldapadd, 23
  ldapdelete, 24
  ldapmodify, 23, 24
  ldapmodrdn, 24
  ldappasswd, 28
  ldapsearch, 24
  slapadd, 21
  slapcat, 22
  slapindex, 22
  slappasswd, 28
  smbldap-groupadd, 76
  smbldap-groupdel, 76
  smbldap-groupmod, 76
  smbldap-passwd, 76
  smbldap-populate, 74
  smbldap-useradd, 75
  smbldap-userdel, 74
  smbldap-usermod, 76
  squid_ldap_group, 78
configurazione
  .ldaprc, 14
  /etc/cpu/cpu.conf, 55
  /etc/ldap/ldap.conf, 13
  /etc/ldap/slapd.conf, 15
  /etc/libnss-ldap.conf, 56
  /etc/migrationtools/migrate_common.ph, 52
  /etc/pam_ldap.conf, 60

demone
  slapd, 11
  slurpd, 42
Distinguished Name, 4, 5, 7, 13, 14, 20, 24, 25, 27, 29, 30, 56, 65, 66

filtri di ricerca, 7, 20, 25–27, 29

giornale di replicazione, 43

Name Service Switch, 1, 50, 53, 56–58

OpenLDAP
  direttiva client
    BASE, 14
    BINDDN, 13
    HOST, 13
    PORT, 13
    SIZELIMIT, 13
    TIMELIMIT, 13
    TLS_CACERT, 38
    TLS_CERT, 38
    TLS_CHECKPEER, 38
    TLS_KEY, 38
    TLS_REQCERT, 38
    URI, 14
    bind_policy, 56
    debug, 13
    idle_timelimit, 56
    nss_base_<map>, 56
    nss_connect_policy, 56
    nss_map_attribute, 56
    nss_map_objectclass, 56
    pam_login_attribute, 60
    pam_max_uid, 60
    pam_min_uid, 60
    pam_password, 60
    rootbinddn, 13
    scope, 13
  direttiva server
    TLSCACertificateFile, 37
    TLSCertificateFile, 37
    TLSCertificateKeyFile, 37
    TLSVerifyClient, 37
    access, 28

`allow`, 18
`argsfile`, 18
`backend`, 15
`cachesize`, 19
`checkpoint`, 19
`database`, 15
`directory`, 19
`disallow`, 18
`idletimeout`, 18
`include`, 17
`index`, 19
`lastmod`, 20
`loglevel`, 18
`mode`, 19
`moduleload`, 18
`modulepath`, 18
`overlay`, 40
`pidfile`, 18
`readonly`, 20
`replicationinterval`, 43
`replica`, 43
`replogfile`, 43
`rootdn`, 27
`rootpw`, 27
`schemacheck`, 18
`sizelimit`, 17
`suffix`, 20
`syncprov-checkpoint`, 46
`syncprov-sessionlog`, 46
`syncrepl`, 47
`timelimit`, 18
`updatedn`, 44
`updateref`, 44

proxy replication, 49–50

referral, 8, 41

Security ID, 69, 72, 73, 76
suffisso base, 2, 5, 7, 13, 14, 20, 22, 25, 56, 57, 67, 68, 70
suffisso base, 73, 77

Bibliografia

[AGL] Simone Piccardi. *Amministrare GNU/Linux*, volume 1. http://labs.truelite.it, 2003.

[RegExp] Jeffrey E. F. Friedl. *Mastering regular expression.* O'Reilly, 1997.

[SGL] Simone Piccardi. *La sicurezza con GNU/Linux*, volume 1. http://labs.truelite.it, 2003.

[WebServ] Simone Piccardi. *I servizi web*, volume 1. http://labs.truelite.it, 2003.