

Sistemi di autenticazione *Protocollo LDAP*

Materiale preparato da Fabio Bucciarelli - DEIS

L'esigenza

- **Ancora ...**
- L'approccio a file diventa inefficiente
- Occorre trovare un database centralizzato e in particolare un servizio centralizzato di autenticazione
- **Nel corso del tempo sono state pensate diverse soluzioni:**
- Nis
- Nis gateway
- Pam
- Ldap

3

L'esigenza

- **Un sistema correttamente funzionante dipende da parecchi file di configurazione**
- **Ma ...**
- In una rete possono esserci centinaia di macchine che hanno gli stessi file di configurazione
- Aggiungere un utente ,aggiungere un host, vuol dire cambiare centinaia di file di configurazione
- **Allora ...**
- Nasce l'esigenza di condividere queste informazioni tra sistemi diversi

2

NIS

- Basato su RPC
- Rilasciato da SUN negli anni '80
- Uno o più server su cui gira ypbind
- Nei file passwd/ shadow/ group una linea con + che indica al sistema operativo che il file non contiene tutto, ma che occorre interrogare NIS
- **Problemi:** complicato, non trasparente alle applicazioni, non esportabile fuori da unix, **NON FLESSIBILE**
- Oramai solo storico

4

NIS gateway

- **Separazione fra le sorgenti di informazione e le altre componenti del sistema**
 - Il file /etc/nsswitch.conf indica per ogni risorsa le fonti da consultare e con che ordine
 - Es.
 - passwd: files nis ldap
 - Group: files ldap
 - hosts: ldap [NOTFOUND=return] dns files

5

PAM (Pluggable Authentication Modules)

- **E' una serie di shared library che permettono all'amministratore di scegliere quali applicazioni usare per l'autenticazione**
 - Scopo di PAM è di separare lo strato di software applicativo da quello dagli appropriati schemi di autenticazione
 - Le applicazioni che devono autenticare un utente possono usare la libreria di funzioni fornita con PAM
 - Chiunque può scrivere i propri moduli PAM
 - La configurazione di PAM attraverso il file /etc/pam.d oppure una serie di file di configurazione nella directory /etc/pam.d

7

NSS

- `<entry> ::= <database> ":" [<source> [<criteria>]]*`
- `<criteria> ::= "[" <criteron> + "]"`
- `<criteron> ::= <status> "=" <action>`
- `<status> ::= "success" | "notfound" | "unavail" | "tryagain"`
- `<action> ::= "return" | "continue"`
- `<action> ::= "return" | "continue" | "forever" | <n>`
- `<n> ::= 0...MAX_INT`
- **Nascita di un'interfaccia locale standard**
- **La fonte xxx è implementata dalla libreria nssxxx.o: chiunque può scrivere la propria fonte**

6

Un altro passo

- **PAM risolve i problemi di standardizzazione di interfacce in locale. Non risolve il problema della complessità di una gestione per migliaia di utenti**
- **Primo approccio:**
 - Scrivere una libreria nss o un modulo PAM per accedere ad un database
- **Problemi:**
 - Ogni sistemista scrive la sue librerie: incompatibilità, difficile testabilità e manutenzione

8

Un altro passo

- **Secondo approccio**
- Standardizzare il protocollo di accesso ad una base dati standardizzata
- **LDAP e directory service**
- Utilizzo di nssldap
- Utilizzo di pam_ldap

9

LDAP

- **Cosa è directory service**
- Lightweight Directory Access Protocol
- Mutuata da X.500 (OSI)
- Directory Services (RFC 1777)
- Molte letture poche scritture (elenco del telefono)
- Né transizioni né rollback
- Replica dei dati

11

E per quel che riguarda Windows?

- **A partire da Windows 2000 è presente la tecnologia Active Directory**
- Alla base della tecnologia Active Directory c'è un directory server
- Active Directory contiene tutte le informazioni riguardanti la macchina stessa e l'unità organizzativa di cui fa parte
- **Si apre uno scenario interessante: poter gestire tutte le informazioni da un'unica interfaccia che utilizzi il protocollo LDAP**
- In ambienti misti Unix / Windows è possibile avere un repository centralizzato e comune degli utenti
- Es: i nostri laboratori

10

LDAP

- Architettura master – slave
- Consistenza fra le copie lasca
- Possibilità di avere amministratori locali
- Gli oggetti di dati sono relativamente piccoli
- L'informazione è basata su attributi
- La ricerca è un'operazione comune

12

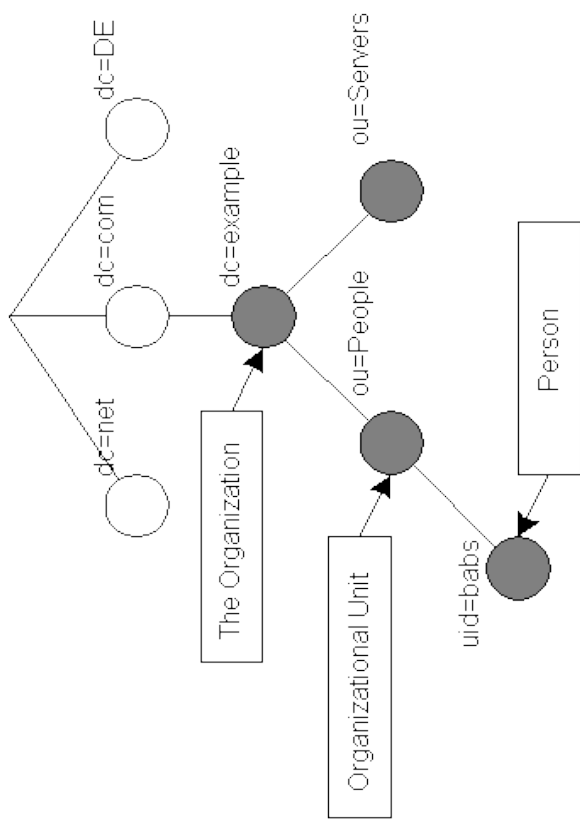
LDAP

• DBI & DIT

- La base di dati della DIR è detta DBI (Directory Information Base)
- DBI è una serie di **entries** (o objects)
- Le entries consistono in una serie di **attributes**
- Gli attributi consistono di **types** con (molteplici) **values**
- Ogni entry ha un DN (Distinguish name)
- Le entries sono gerarchicamente legate. La gerarchia è chiamata DIT (Dir Information Tree). Una DBI è rappresentata in un DIT.

13

Dir Name Space



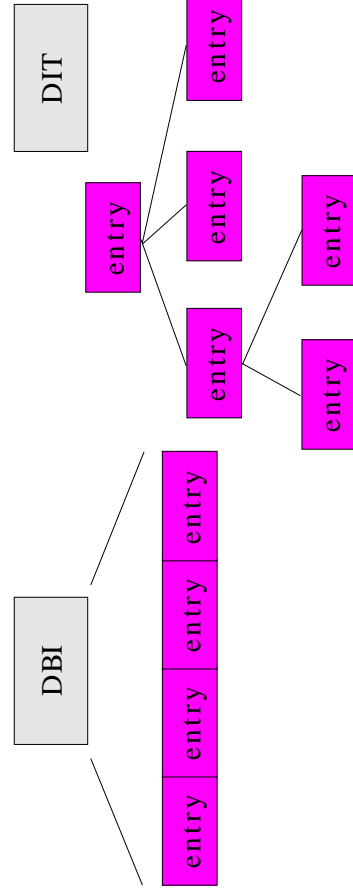
15

LDAP

• Scopi

- Riferimento ai dati
- Organizzazione dei dati
- Partizionamento dei dati
- Replica dei dati
- Controllo degli accessi
- Supporto alle applicazioni

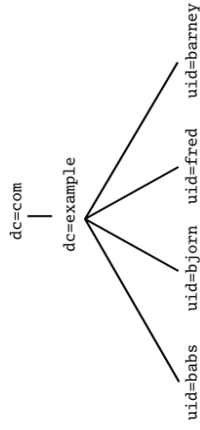
14



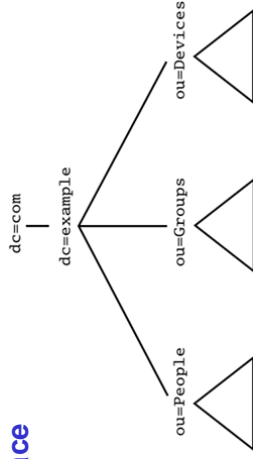
16

Dir Name Space

- **Flat namespace**



- **Hierarchical namespace**

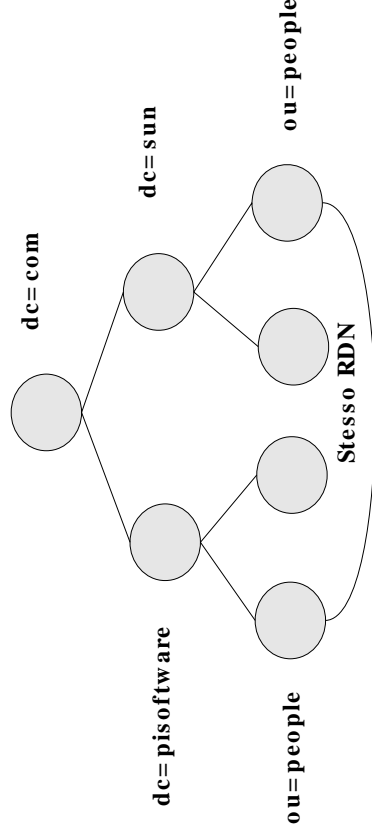


17

LDAP

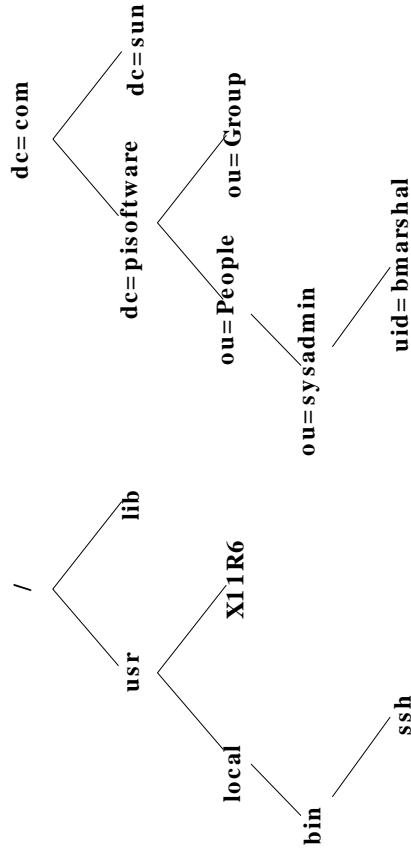
- **Distinguished Name**

- Elenco degli attributi separati da virgole a partire dal fondo
- DN = BDN (Base DN) + RDN (Relative DN)



19

Confronto con un file system



Nel FS esplorazione dalla radice, nella DIR dalle foglie

18

DN: esempio

- DN: uid=bmarshal,ou=People,dc=software,dc=com
- RDN: uid=bmarshal
- BDN: ou=People,dc=software,dc=com
- Ogni RDN distingue le entries fra i pari
- In ogni BDN ogni RDN è unico: **ogni entries ha un DN unico**
- Ogni RDN può essere composto dalla concatenazione di più attributi (per renderlo unico)
- In ogni entry è specificato quali attributi ne costituiscono il RDN

20

Il protocollo LDAP

- **Usa il protocollo TCP/IP in una maniera molto efficiente**
- **Esempio di interazione client / server**
 - Il client si connette al server e richiede un'operazione di bind
 - Il server restituisce un codice di successo o chiude la connessione
 - Il client richiede un'operazione di search (o un'altra operazione)
 - Il server restituisce un messaggio con le entry trovate o niente se niente è stato trovato
 - Il server restituisce un codice a seconda del risultato dell'operazione
 - Il client richiede un'operazione di unbind

21

Filtri (RFC 1558)

- **Selezione sugli attributi**

```
<filter> ::= '(' <filtercomp> ')'  
<filtercomp> ::= <and> | <or> | <not> | <item>  
<and> ::= '&' <filterlist>  
<or> ::= '|' <filterlist>  
<not> ::= '!' <filter>  
<filterlist> ::= <filter> | <filter> <filterlist>  
<item> ::= <simple> | <present> | <substring>  
<simple> ::= <attr> <filtertype> <value>  
<filtertype> ::= <equal> | <approx> | <greater> | <less>  
<equal> ::= '='  
<approx> ::= '~'  
<greater> ::= '>'  
<less> ::= '<'  
<present> ::= <attr> '=' *  
<substring> ::= <attr> '=' '!' <initial> <any>  
<final>  
<initial> ::= NULL | <value>  
<any> ::= '*' <starval>  
<starval> ::= NULL | <value> '*' <starval>  
<final> ::= NULL | <value>
```

23

L'operazione di search

- **Consiste di una richiesta del client, di un lavoro da parte del server e di un risultato**
- **Alcuni parametri specificano cosa cercare e come cercare**
 - Base DN
 - Bind DN
 - Uno scope
 - base
 - one
 - subtree
 - Filtro

22

Filtri

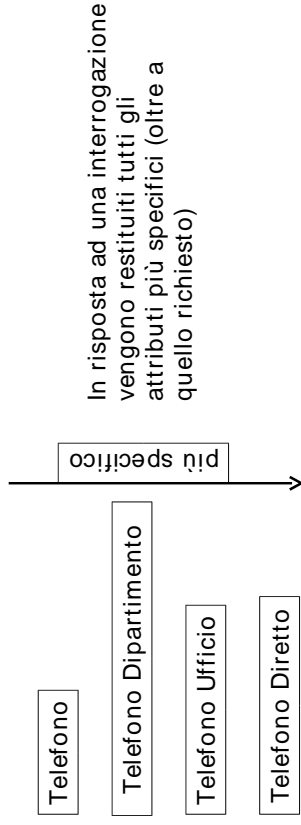
- **Esempi di filtri**

```
(cn = Babs Jensen)  
!(cn = Tim Howes)  
(&  
  (objectClass= Person)  
  (sn = Jensen)(cn = Babs J*)  
)  
(o = univ*of*mich*)  
(objectclass= posixAccount)  
(&((uid = jack)(uid = jill))(objectclass= posixAccount))
```

24

Attributi in gerarchia

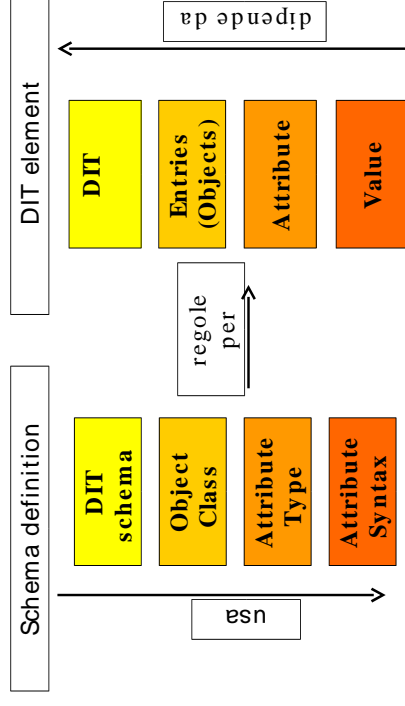
- Sono possibili gerarchie di attributi



25

Schema

- Il modo per descrivere le parti del sistema è detto schema e la sintassi usata è ASN1



27

Schema

- L'insieme delle regole che descrivono i dati immagazzinati
- L'uso di schemi rende chiaro la struttura delle Dir
- Per ogni entries (object) lo schema indica la regola da seguire per memorizzare i dati indicando:

- Attributi indispensabili / facoltativi
- Come comparare gli attributi
- Il tipo di dato memorizzabile in un attributo

26

Esempi di attributes

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

```
attributetype ( 1.3.6.1.1.1.1.4 NAME 'loginShell'  
DESC 'The path to the login shell'  
EQUALITY caseExactIA5Match  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

28

Schema

- **Alcuni attributi standard**

uid	User id
cn	Common name
sn	Surname
l	Location
c	Country

ou	Organization Unit
o	organization
dc	Domain Component
st	State

29

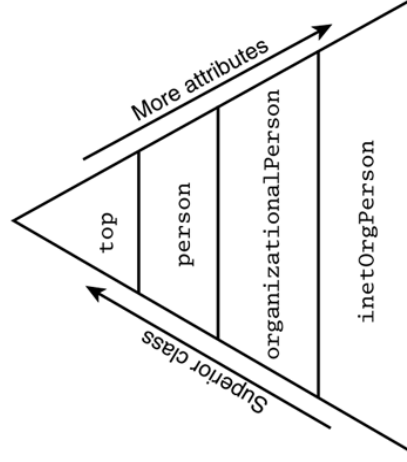
Esempi di object class

```
objectclass(
  1.3.6.1.1.1.2.0 NAME 'posixAccount' SUP top AUXILIARY
  DESC 'Abstraction of an account with POSIX attributes'
  MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
  MAY ( userPassword $ loginShell $ gecos $ description )
)
objectClass (
  1.3.6.1.4.1.4203.1.4.5 NAME 'OpenLDAPperson'
  DESC 'OpenLDAP Person'
  SUP ( pilotPerson $ inetOrgPerson )
  MUST ( uid $ cn )
  MAY ( givenName $ labeledURI $ o )
)
```

31

Schema

- **Ereditarietà dell'object class**



30

Operazioni su ldap

- **Search**
- **Compare**
- **Add**
- **Delete**
- **Modify**
- **ModifyRDN**
 - Rinomina una entry o sposta la entry all'interno della directory
- **Strumenti a riga di comando: ldapsearch, ldapadd, ldapmodify, ...**

32

LDAP URL (RFC 1959)

```
<ldapurl> ::= "ldap://" [ <hostport> ] "/" <dn> [ "?" <attributes> [ "?"
<scope> "?" <filter> ] ]
<hostport> ::= <hostname> [ ":" <portnumber> ]
<dn> ::= a string (RFC 1485)
<attributes> ::= NULL | <attributelist>
<attributelist> ::= <attribute> | <attributetype> [ "," <attributelis<> ]
<attribute> ::= a string (RFC 1777)
<scope> ::= "base" | "one" | "sub"
<filter> ::= a string (RFC 1558)
ldap://foo.bar.com/dc=bar,dc=com
ldap://argle.bargle.com/dc=bar,dc=com??sub?uid=barn
ey
ldap://ldap.bedrock.com/dc=bar,dc=com?cn?sub?uid=b
arney
```

33

LDIF

• Esempio

```
dn:
uid=lab2n36$,ou=People,o=People,ou=labx,dc=ing,dc=unibo,dc=it
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
objectClass: top
userPassword: e2NSeX28fXg=
loginShell: /bin/false
uidNumber: 62036
gidNumber: 0
homeDirectory: /dev/null
gecos: Macchina windows
uid: lab2n36$
cn: LAB2N36
```

35

LDIF

• Formato di scambio ASCII (LDAP Interchange Format)

- Entries rappresentate in ascii
- Umanamente leggibili
- Permette facilmente di trasferire / modificare i dati

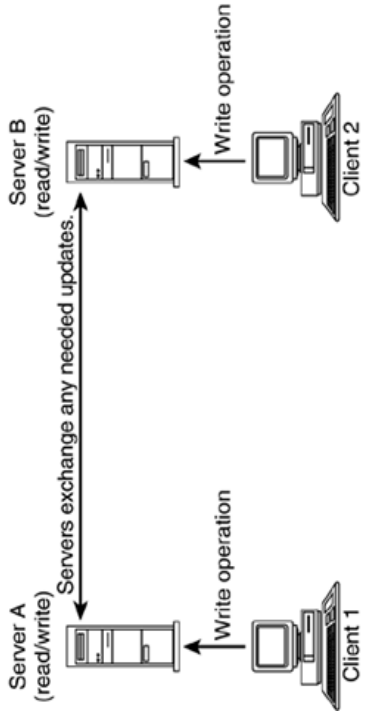
34

Replica

- **Avere repliche: fault tolerance, performance**
- LDAP ha un'architettura master – replica: le modifiche vengono propagate dal master alle repliche
- Lo standard non dice con che tempistica né in che modo
- Lo standard attuale in sviluppo (v3) prevede un'architettura multimaster (protocolli ad hoc per la replica)
- Nel caso monomaster più modalità per gestire la scrittura delle entries (on master, by referral, by chain)

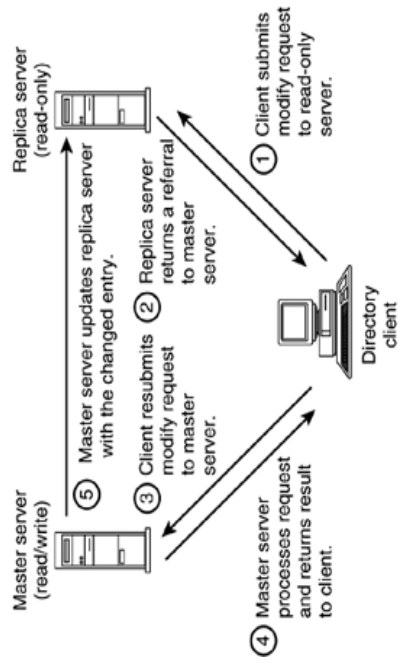
36

Modello multimaster



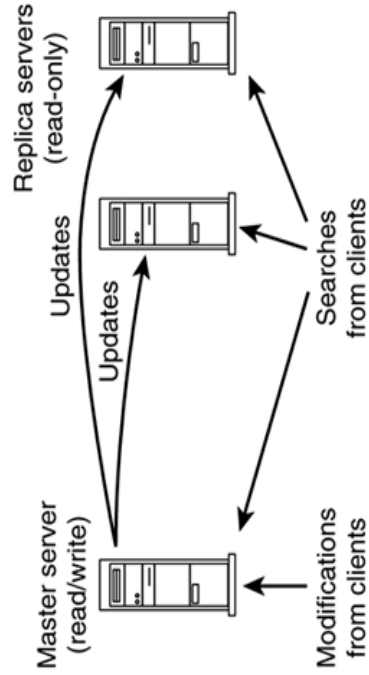
37

By referrals



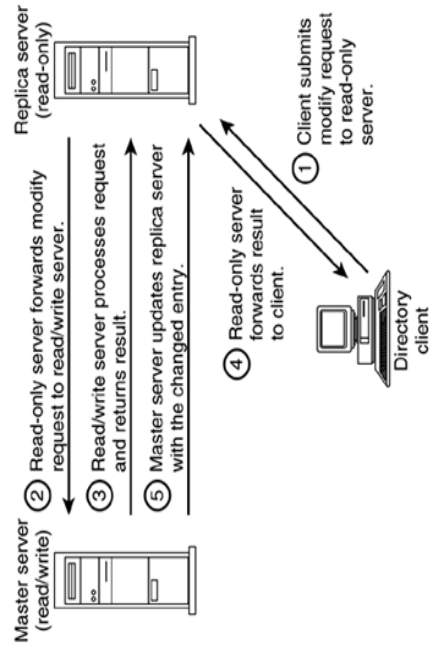
39

On master



38

By chain



40

Directory distribuite

- **Un sottoalbero della directory viene delegato ad un altro server**

- Performance
- Locazione geografica
- Delega amministrativa
- Occorrono due link: uno sul lato del server principale e uno sul lato del server che ottiene la delega

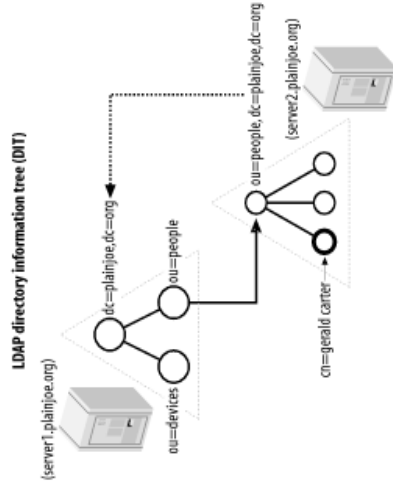
41

Directory distribuite

- **Superior knowledge link**
- Contiene l'URI del server a cui ci si riferisce per la parte superiore della directory
- **Subordinate knowledge link**
- Chiamato semplicemente **reference**, connette logicamente un nodo all'interno di un albero al context naming di un altro server
- Nell'esempio precedente, la entry `ou=people,dc=plainjoe,dc=org` contiene un referral al server a cui si delega il sottoalbero

43

Directory distribuite



42

Directory distribuite

- **L'object class referral è definito**
-
- The diagram shows a box labeled `objectClass: referral` with a `ref:` attribute. A line connects the `ref:` attribute to a list of required attributes: `(2.16.840.1.113730.3.2.6 NAME 'referral' DESC 'named subordinate reference object' STRUCTURAL MUST REF)`.
- **L'LDIF della entry nella figura precedente è definita:**
 - `dn: ou=people,dc=plainjoe,dc=org`
`objectClass: referral`
`ref: ldap://server2.plainjoe.org/ou=people,dc=plainjoe,dc=org`

44

Alcune implementazioni di LDAP

- **Tre principali prodotti**
- OpenLDAP
- Microsoft Active Directory
- Directory Server

45

Active Directory

- **Realizzato da Microsoft, che, a partire da Windows 2000, ne ha fatto la base dei propri sistemi server**
- Strettamente legato a Windows
- Facile da gestire e amministrare
- Buona la scalabilità
- Active Directory Application Mode (“promessa” di una completa libertà nel design del proprio namespace e nella definizione di schemi personalizzati)

47

OpenLDAP

- **Realizzato dall'Università del Michigan, secondo la filosofia Open Source**
- E' gratuito e l'accesso al codice sorgente è libero
- Può “girare” su qualsiasi piattaforma
- La comunità degli sviluppatori e degli utilizzatori è molto attiva
- La possibilità di controllo sugli accessi è imponente
- Implementa la replica solo attraverso il modello monomaster, la possibilità di Directory distribuite è stata introdotta solo nelle ultime versioni, mentre mancano altre possibilità più evolute

46

Directory Server

- **Realizzato da Netscape, è il primo Directory Server commerciale apparso sul mercato**
- Cambiato molte volte, ora in due versioni molto simili:
 - Netscape Directory Server
 - Sun ONE Directory Server
- Diversi prodotti si integrano con Directory Server
- Disponibile per tutte le principali piattaforme
- Ha le migliori prestazioni sul mercato
- La documentazione è ottima

48