Sistemas Operativos Distribuidos

Servicio de Directorio

Índice

Servicio de directorio

- Punto de acceso es sólo uno de los atributos de una entidad
  - Nombre impresora → modelo, color, ubicación, formatos soportados, ...
  - Pueden gestionarlos servidores específicos
    - Servicio de impresión gestiona información de impresoras
  - Problema: Duplicidad de funcionalidad
- Solución: Generalización del servicio de nombres
  - Nombre → conjunto de atributos de la entidad
- Aplicable a entidades de infraestructura del SD y de "negocio"
  - En FI: alumnos, profesores, títulos, asignaturas, dptos., servicios, ...
- Servicio de directorio (Sdir):
  - Repositorio de información de entidades de SD
    - Sun: "globalización" de */etc* → *Network Information System* (NIS)
  - No todo atributo ⊂ Sdir: no incluir atributos muy dinámicos
    - Tamaño cola de trabajos debería gestionarlo el servicio de impresión

Tipos de resolución de nombres

- Convencional (*Páginas blancas*): nombre → atributos
- Por atributos (*Páginas amarillas*): atributos → entidades
  - "Quiero imprimir fichero en impresora en color cerca de mi despacho"
- Parámetros típicos en resolución por atributos:
  - Nodo de inicio de búsqueda (base)
    - P.e. Sólo buscar entidades en determinada sucursal de la empresa
  - Profundidad de la búsqueda (sub-árbol, hijos directos, sólo base, ...)
  - Criterio/filtro de búsqueda:
    - Función lógica que deben satisfacer las entidades buscadas
    - P.e. Impresoras en color que estén ubicadas en el tercer piso
  - Límite de tiempo de búsqueda
  - Nº máximo de entidades que se retornarán
  - Atributos que se retornarán de las entidades seleccionadas

Tipos de entidades gestionadas

- ¿Qué tipos de entidades gestiona un servidor de nombres?
- Predeterminado:
  - Tipos de entidades predefinidas
- Configurable:
  - Existe un mecanismo para definir los tipos de las entidades
    - hay que definir: nombre del tipo, atribs., tipos de los atribs., etc.
  - Separación entre definición de tipos de entidades y de entidades
    - Similitud con base de datos: esquemas y datos
    - Similitud con POO: clases y objetos
  - Extensible: tipos predefinidos pero se pueden definir adicionales
    - Puede ser útil la herencia (simple o múltiple)

Serv. directorio vs. Base de datos

- Hay alguna similitud
  - Repositorio de información
  - Permite búsqueda sofisticada
- Pero muchas diferencias. Servicio de directorio:
  - Muchas consultas pero muy pocas modificaciones
  - Transacciones muy simples
  - Uso de esquemas estándar, siempre que sea posible
  - Más facilidad para cambiar esquemas para datos ya creados
    - P.e. Añadir a profesor FI asignaturas que imparte
  - Más adecuado para datos jerárquicos
  - Datos con múltiples valores para cada atributo
  - Si datos replicados, no requiere coherencia estricta
- Aunque muchos Sdir implementados con una base de datos

Lightweight Directory Access Protocol

- Precedente: X.500 servicio de directorio de ISO
  - Concebido para ser un directorio mundial
  - Complejo
  - Pesado: Ejecuta sobre la pila OSI
  - Protocolo de acceso DAP (*Directory Access Protocol*)
- LDAP (*Lightweight Directory Access Protocol*, RFC 4510)
  - Basado en X.500
  - Más sencillo
  - Más ligero: ejecuta sobre la pila TCP/IP
  - Es un protocolo pero define implícitamente un modelo de datos
    - No define aspectos de implementación
  - Distintos sistemas ofrecen una interfaz LDAP (p.e. *Active Directory*)
  - Actualmente versión 3

Objetos y clases

- Entidad $\rightarrow$ Objeto (entrada) en LDAP
  - Orientado a objetos: Objeto $\in$ Clase (atributo *objectClass*)
- Clase define conjunto de atributos del objeto
  - Tipo del atributo | obligatorio(ob) u optativo(op) | valor único o múltiple
- Herencia: clases forman una jerarquía (*top* raíz de jerarquía)
  - Clase derivada hereda atributos de superclases
- Tipos de clases:
  - Abstracta (AB): no pueden definirse objetos de esa clase (p.e. *top*)
  - Estructural (ES): Objeto $\in$ Una y solo una clase estructural
    - No puede cambiar la clase estructural de un objeto
  - Auxiliar (AU): Objeto puede estar asociado a varias clases auxiliares
    - Pueden añadirse dinámicamente: Facilitan extensión de objetos
  - Superclase(ES)=ES|AB; Superclase(AU)=AU|AB

Ejemplos de clases

- *top*: raíz; AB; ob: *objectClass*
- *person*: ↓*top;* ES; ob: *cn*, *sn*; op: *telephoneNumber*, ...
- *residentialPerson*: ↓*person;* ES; ob: *l;* op: *postalAddress*, ...
- *organization*: ↓*top;* ES; ob: *o;* op: *postalAddress*, ...
- *organizationalUnit*: ↓*top;* ES; ob: *ou;* op: *postalAddress*, ...
- *dcObject*: ↓*top;* AU; ob: *dc* (valor único)
- *device*: ↓*top;* ES; ob: *cn*; op: *serialNumber, o, ou, owner*, ...
- *groupOfNames*: ↓*top;* ES; ob: *cn*, *member*; op: *o, ou*, ...
- *alias*: ↓*top;* ES; ob: *aliasedObjectName*
- *referral*: ↓*top;* ES; ob: *ref*

Extracto de mi entrada en LDAP de FI

Formato de texto LDIF (*LDAP Data Interchange Format*): protocolo LDAP es binario

objectClass: inetOrgPerson ← estructural (*top*→*person*→*organizationalPerson*→*inetOrgPerson*)
objectClass: posixAccount ← auxiliar (*top*→ *posixAccount*)
objectClass: fiEmployee ← auxiliar (*top* → *irisPerson* → *fiPerson* → *fiEmployee*)
objectClass: sambaSamAccount ← auxiliar (*top* → *sambaSamAccount*)
cn: Fernando Perez Costoya
cn: F. P. Costoya
sn: Perez Costoya
telephoneNumber: 913367377
mail: fperez@fi.upm.es
uid: fperez
person / *organizationalPerson* / *inetOrgPerson*
uidnumber: ....
gidNumber: ....
posixAccount
irisUserStatus: Activo
fiRelationShip: pdi
fiTeaching: .....
irisPerson / *fiPerson* / *fiEmployee*
sambaSID: .....
sambaSamAccount

Extracto de entrada FI en LDAP de FI

objectClass: dcObject ← auxiliar (*top* → *dcObject*)
objectClass: organization ← estructural (*top* → *organization*)
objectClass: labeledURIObject ← auxiliar (*top* → *labeledURIObject*)
dc: fi ← atributo específico de *dcObject*
o:: RmFjdWx0YWQgZGUgSW5mb3Jtw6F0aWNhIC0gVVBN
postalCode: 28660
l: Boadilla del Monte
st: Madrid
labeledURI: http://www.fi.upm.es ← atributo específico de *labeledURIObject*
telephoneNumber: +34 913367399

Decodificación de base 64
o: Facultad de Informática – UPM

Modelo de nombres

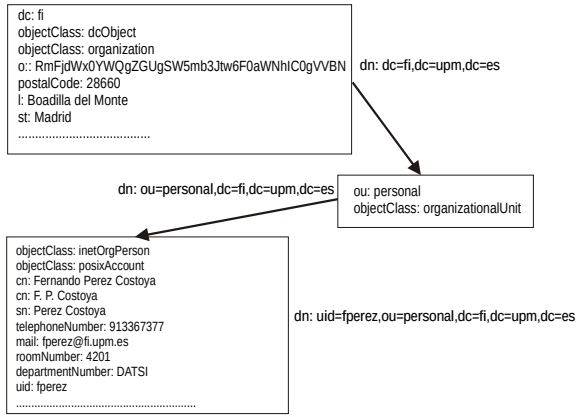
- Entrada tiene un nombre: *Relative Distinguished Name* (RDN)
  - 1 o más atributos de la entrada que la hacen única entre "hermanos"
    - *uid=fperez* (ej. múltiples*: cn=Fernando Perez Costoya+dni=76543210)*
- Jerarquía de nombres (*Directory Information Tree*, DIT)
  - Nombre completo (*path*): *Distinguished Name* (DN)
    - RDN de la entrada + DN del padre (separados por comas)
      - ***dn:*** *uid=fperez,ou=personal,dc=fi,dc=upm,dc=es*
    - No confundir con jerarquía de clases
    - Similar a SF pero directorios también tienen información asociada
  - Nombre del objeto raíz (sufijo o base): a discreción
    - Convenio: a partir de dominio DNS usando clase auxiliar *dcObject*
      - Dominio: *fi.upm.es* → ***dn:*** *dc=fi,dc=upm,dc=es*
  - Servidor LDAP gestiona 1 ó más DIT
  - Servidor devuelve metainformación en objetos/atrib. operacionales
    - DIT gestionados por el servidor, esquemas soportados, ...

Extracto de rama del DIT del LDAP de FI

```
# fi.upm.es
dn: dc=fi,dc=upm,dc=es
dc: fi
objectClass: dcObject
objectClass: organization
.....
# personal, fi.upm.es
dn: ou=personal,dc=fi,dc=upm,dc=es
ou: personal
objectClass: organizationalUnit

dn: uid=fperez,ou=personal,dc=fi,dc=upm,dc=es
uid: fperez
.....
```


Extracto de rama del DIT del LDAP de FI



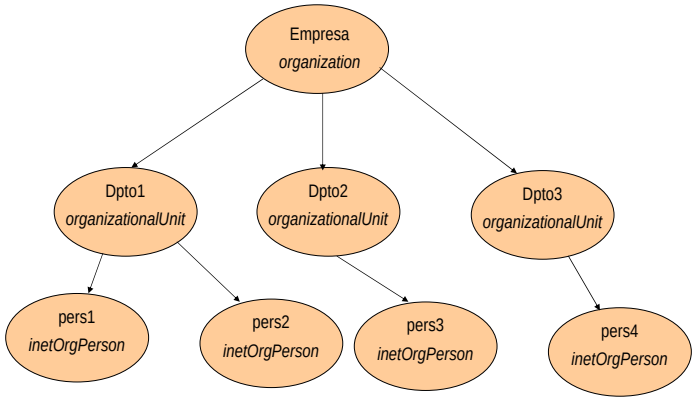
Distribución y replicación

- Espacio de nombres distribuido usando *referrals*
  - Objeto en DIT especifica punto de montaje
  - No definido el modelo de navegación
    - Implementación más habitual iterativa
    - Aunque también recursiva (*chaining*)
- Replicación de espacio de nombres no definida por estándar
  - OpenLDAP admite dos esquemas:
    - Maestro-esclavo: asimétrico
    - Multi-maestro: simétrico
  - OpenLDAP no garantiza coherencia

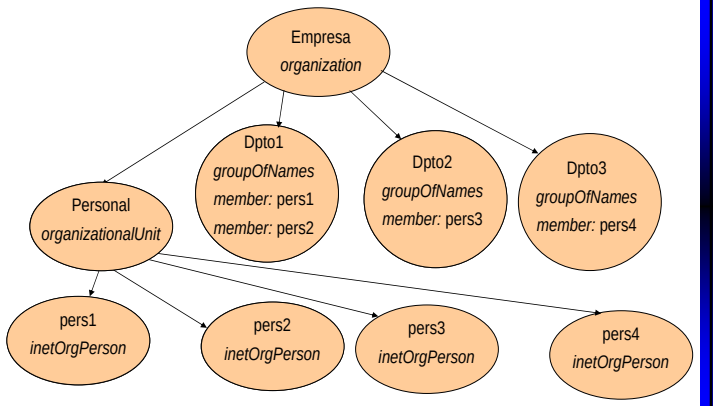
Diseño del DIT

- No trivial: requiere experiencia
- Análisis previo de info. del SD y cómo evolucionará
  - Diseño debería evitar que cambios previstos en info. modifiquen DIT
    - Cambio debería afectar a atributos en vez de a estructura de DIT
  - Mejor árbol poco profundo
- Ej.: empresa donde personal cambia de dpto. con frecuencia
  - Diseño 1
    - 1 *organizationalUnit*/dpto. + 1 *inetOrgPerson*/persona
    - Entrada de persona hija de entrada de su departamento
  - Diseño 2
    - 1 *organizationalUnit* para todo el personal + 1 *inetOrgPerson*/persona
    - 1 *groupOfNames*/dpto. con 1 atributo *member*/persona
    - Persona cambia de departamento: cambio atributos, no cambio DIT
      - Aunque ciertas búsquedas pueden ralentizarse

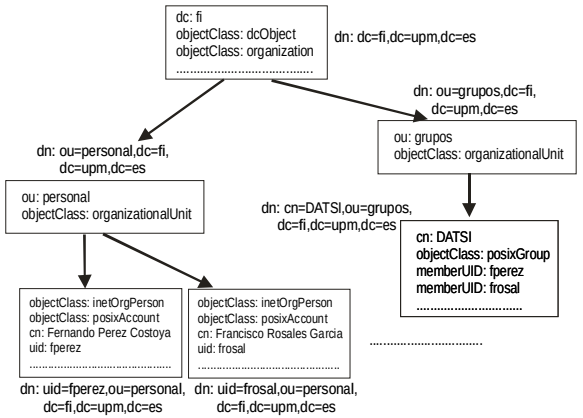
Diseño 1



Diseño 2



Extracto de jerarquía de LDAP de FI



Operaciones de LDAP

- *Bind/Unbind*: conecta y autentica/desconecta
- *Search*: realiza una búsqueda basada en los parámetros:
  - DN base de la búsqueda
  - Ámbito: Sólo la entrada base, sólo hijos o todo el sub-árbol
  - Filtro de búsqueda
  - Atributos que se devuelven (además, si valores o sólo tipos)
  - Si se siguen los alias o no durante la búsqueda
  - Límite de tiempo y máximo nº de entradas retornadas
- *Compare*: comprueba si DN dado tiene un valor en atributo
- *Add/Delete*: Añade/Elimina la entrada del DN dado
- *Modify*: Modifica atributos (añade, elimina o cambia) de un DN
- *Modify DN*: Cambia DN de una entrada
  - Renombra si sólo cambia RDN final; mueve en DIT en caso contrario

Acceso a operaciones de LDAP

- API de programación en C
  - *ldap_bind()*,*ldap_search()*,*ldap_add()*,*ldap_delete()*,*ldap_modify()*, ...
- Mandatos
  - *ldapsearch*, *ldapadd*, *ldapdelete*, *ldapmodify*, *ldapmodrdn*, ...
    - La mayoría usan el formato LDIF como entrada o salida
- Formato URL estándar para LDAP
  - *ldap://máquina:puerto/DNbase?atributos?ámbito?filtro*
    - *ldaps* si usa comunicación segura

Ejemplos de búsquedas (en triqui)

- Leer mi entrada

```
ldapsearch -x -W -H ldaps://info.fi.upm.es -D 'uid=fperez,ou=personal,dc=fi,dc=upm,dc=es'
    -b 'uid=fperez,ou=personal,dc=fi,dc=upm,dc=es'
```

- Nombre de profesores que comparten un despacho dado

```
ldapsearch  -x -W -H ldaps://info.fi.upm.es -D 'uid=fperez,ou=personal,dc=fi,dc=upm,dc=es'
    -b 'ou=personal,dc=fi,dc=upm,dc=es'  '(roomNumber=4201)' cn sn
```

- Nº tel. de personal de nombre Fernando y no sean del DATSI

```
ldapsearch  -x -W -H ldaps://info.fi.upm.es -D 'uid=fperez,ou=personal,dc=fi,dc=upm,dc=es'
    -b 'ou=personal,dc=fi,dc=upm,dc=es' '(&(!(departmentNumber=DATSI))(cn=*Fernando*))' cn
    telephoneNumber
```

- Nombre de “secciones” de la FI

```
ldapsearch -x -W -H ldaps://info.fi.upm.es -D 'uid=fperez,ou=personal,dc=fi,dc=upm,dc=es'
    -b 'dc=fi,dc=upm,dc=es' -s one '(objectClass=organizationalUnit)' ou
```


Esquema

- Paquete que incluye definiciones en ASN.1 y que usan OIDs
- Esquema incluye varios tipos de definiciones:
  - *ldapsyntax*: Define tipos básicos de LDAP
  - *matchingRule*: Op. de comparación sobre tipos básicos
  - *attributetype*: Definición de atributo
  - *objectclass*: Definición de clase
  - *matchingRuleUse*: Para qué atributo se usa una regla de comparación
  - *dITContentRule*: qué clases auxiliares permitidas para una c. estruct.
  - *dITStructureRule*: qué clases pueden ser padres de una c. estructural
  - *nameForm*: qué atributos pueden usarse como RDN de c. estructural
- Se usa herencia tanto en defs. de clases como de atributos
- Hay esquemas estandarizados:
  - *core*, *cosine*, *inetorgperson*, *nis*, ...

Sintaxis: tipos de datos de LDAP

- Definidos por estándar, por interoperabilidad no deberían definirse nuevos tipos

```
ldapsearch -H ldaps://info.fi.upm.es -x  -b cn=subschema -s base ldapsyntaxes

ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.44 DESC 'Printable String' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.11 DESC 'Country String' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.26 DESC 'IA5 String' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.40 DESC 'Octet String' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.41 DESC 'Postal Address' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.50 DESC 'Telephone Number' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.36 DESC 'Numeric String' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.27 DESC 'Integer' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.24 DESC 'Generalized Time' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.7 DESC 'Boolean' )
ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.6 DESC 'Bit String' )
……
```

Reglas de comparación de tipos

- Definidas por estándar, por interoperabilidad no deberían definirse nuevas reglas

ldapsearch -H ldaps://info.fi.upm.es -x -b cn=subschema -s base matchingRules

matchingRules: (2.5.13.4 NAME 'caseIgnoreSubstringsMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.58)
matchingRules: (2.5.13.2 NAME 'caseIgnoreMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
matchingRules: (1.3.6.1.4.1.1466.109.114.3 NAME 'caseIgnoreIA5SubstringsMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
matchingRules: (1.3.6.1.4.1.1466.109.114.2 NAME 'caseIgnoreIA5Match' SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
matchingRules: (2.5.13.14 NAME 'integerMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27)
matchingRules: (2.5.13.13 NAME 'booleanMatch' SYNTAX 1.3.6.1.4.1.1466.115.121.1.7)
……

Definición de atributos

ldapsearch -H ldaps://info.fi.upm.es -x -b cn=subschema -s base attributetypes

attributetype (2.5.4.41 NAME 'name'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768})

attributetype (2.5.4.3 NAME ('cn' 'commonName') SUP name)

attributetype (0.9.2342.19200300.100.1.25
NAME ('dc' 'domainComponent')
DESC 'RFC1274/2247: domain component'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)

Definición de clases

ldapsearch -H ldaps://info.fi.upm.es -x -b cn=subschema -s base objectClasses

objectclass (2.5.6.0 NAME 'top' ABSTRACT MUST objectClass)

objectclass (2.5.6.6 NAME 'person' SUP top STRUCTURAL
MUST (sn $ cn)
MAY (userPassword $ telephoneNumber $ seeAlso $ description))

objectclass (2.5.6.7 NAME 'organizationalPerson' SUP person STRUCTURAL
MAY (title $ x121Address $ registeredAddress $ destinationIndicator $
preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $
telephoneNumber $ internationaliSDNNumber $
facsimileTelephoneNumber $ street $ postOfficeBox $ postalCode $
postalAddress $ physicalDeliveryOfficeName $ ou $ st $ l))

objectclass (1.3.6.1.4.1.1466.344 NAME 'dcObject'
DESC 'RFC2247: domain component object'
SUP top AUXILIARY MUST dc)

Uso de reglas de comparación

ldapsearch -H ldaps://info.fi.upm.es -x -b cn=subschema -s base matchingRulesUse

matchingRuleUse: (1.3.6.1.4.1.1466.109.114.2 NAME 'caseIgnoreIA5Match' APPLIES (altServer $ mail $ dc $ associatedDomain $ email $ aRecord $ mDRecord $ mXRecord $ nSRecord $ sOARecord $ cNAMERecord $ janetMailbox $ gecos $ homeDir....))
matchingRuleUse: (2.5.13.13 NAME 'booleanMatch' APPLIES (hasSubordinates $ olcGentleHUP $ olcLastMod $ olcReadOnly $ olcReverseLookup $ olcDbNoSync $ olcDbDirtyRead $ olcDbLinearIndex $ olcChainCacheURI $ olcChainReturnError $ olcDbRebindAsUser $ olcDbChaseReferrals $ olcDbProxyWhoAmI $ olcDbSingleConn $ olcDbUseTemporaryConn $ pwdLockout $ pwdMustChange $ pwdAllowUserChange $ pwdSafeModify $ sambaBoolOption $ pwdReset $ olcPPolicyHashCleartext $ olcPPolicyUseLockout $ olcSpNoPresent $ olcSpReloadHint))

Creación de un nuevo esquema

- Sólo si es estrictamente necesario
  - Nunca cambiar comportamiento de objetos/atrib. estándar
- 2 alternativas para extender clase ya existente
  - Crear nueva clase estructural derivada de clase existente
    - Permite mejor control: se pueden definir reglas de contenido/estructura
    - Pero requiere eliminar y reinsertar todos los objetos existentes
  - Crear clase auxiliar derivada de *top* e incluirla en definición de objetos
    - Se puede añadir directamente usando *Modify*
- C. auxiliar también permite incluir atrib. en objetos de ≠ clases
  - p.e. fecha de alta en organización, tanto personas como dispositivos

Extracto de esquema del LDAP de FI

attributetype (1.3.6.1.4.1.7547.1.19.10.4.2.4 NAME 'fiRelationShip' DESC 'Relacion del usuario con la Escuela' EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype (1.3.6.1.4.1.7547.1.19.10.4.2.1 NAME 'fiGender' DESC 'Sexo de la persona (ISO 5218)' EQUALITY integerMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE)

attributeTypes: (1.3.6.1.4.1.7547.1.19.10.4.2.5 NAME 'fiTeaching' DESC 'Asignaturas impartidas por el profesor' EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{20})

objectclass (1.3.6.1.4.1.7547.4.3.1.2 NAME 'irisPerson' DESC 'Persons inside the IRIS community' SUP top AUXILIARY MAY (sn1 $ sn2 $ irisPersonalTitle $ irisPersonalUniqueID $ irisUserEntitlement $ irisUserPrivateAttribute $ irisUserStatus $ irisMailHost $ irisMailRoutingAddress $ irisMailbox $ irisMailMainAddress $ irisMailAlternateAddress $ irisUserPresenceID $ irisClassifCode))

objectclass (1.3.6.1.4.1.7547.1.19.10.4.1.1 NAME 'fiPerson' DESC 'Persona perteneciente a la Facultad de Informatica (UPM)' SUP irisPerson AUXILIARY MUST (uid $ mail) MAY (fiPwdChangedOperTime $ fiMailQuotaSize $ fiGender $fiRelationShip))

objectClasses: (1.3.6.1.4.1.7547.1.19.10.4.1.3 NAME 'fiEmployee' DESC 'Empleado de la Facultad de Informatica (UPM)' SUP fiPerson AUXILIARY MAY fiTeaching)

Modelo de seguridad

- 3 métodos de autenticación
  - Sin autenticación: se considera usuario anónimo
  - Autenticación básica: DN del usuario + contraseña
  - *Simple Authentication and Security Layer* (SASL)
    - Entorno genérico de autenticación y seguridad de datos
    - Permite usar múltiples mecanismos (p.e. SASL DIGEST-MD5)
    - SASL EXTERNAL: protocolo nivel inferior proporciona autenticación
      - Como cuando se usa *Transport Layer Security* (TLS)
- Protección de entradas no definida por el estándar
  - Habitualmente se usan listas de control de acceso (ACL)
    - Controlan acceso a cada atributo de una entrada
- Más sobre estos aspectos en tema de seguridad