Servicios de directorio

Antoni Martínez-Ballesté
Jordi Castellà-Roca

PID_00177509

UOC
Universitat Oberta de Catalunya

Índice

Introducción..... 5

Objetivos..... 6

1. Concepto y uso de los directorios..... 7
1.1. Ejemplos y tipos de directorios 9
1.1.1. Directorios y su uso en seguridad y autenticación 10
1.2. Espacio de nombres 12
1.2.1. Sistema DNS 12
1.2.2. Sistema WINS 14
1.2.3. Sistema LDAP 14
1.3. Operaciones de cliente 16
1.3.1. Operaciones de interrogación 17
1.3.2. Operaciones de actualización 19
1.3.3. Operaciones de autenticación 20
1.3.4. Acceso desde otras aplicaciones 20

2. Diseño del directorio..... 22
2.1. Diseño del espacio de nombres 22
2.1.1. Elección del sufijo 22
2.1.2. Estructura del directorio 23
2.1.3. Identificación de objetos en el directorio 24
2.2. Esquema del directorio 25
2.2.1. Atributo 25
2.2.2. Clase de objeto 28
2.3. Seguridad, eficiencia y disponibilidad del directorio 29
2.3.1. Seguridad en el directorio 29
2.3.2. Eficiencia y disponibilidad 31

3. Implementaciones de servicio de directorio..... 33
3.1. OpenLDAP 33
3.2. *Apache directory server*..... 34
3.3. Active Directory 35

Resumen..... 38

Actividades..... 39

Glosario..... 40

Bibliografía..... 41

Introducción

Uno de los propósitos de los sistemas informáticos es el de guardar información. Los sistemas gestores de bases de datos ofrecen herramientas para guardar ingentes cantidades de información. Esta información puede llegar a tener estructuras realmente complejas, con multitud de entidades y relaciones.

Durante la implantación de los sistemas informáticos, ha sido necesario desarrollar herramientas centradas en gestionar un tipo muy concreto de información. Un ejemplo clásico es la información referente al sistema de ficheros de una unidad de almacenamiento. Se trata de un concepto muy específico, con la información a guardar claramente definida. Ahora bien, es preciso que la implementación sea robusta y escalable, puesto que un sistema de ficheros puede fácilmente llegar a albergar millones.

Los directorios son un tipo específico de bases de datos con un propósito también específico: almacenar la información sobre un objeto (individuo, recurso de red, documento). Su papel es clave en cualquier organización que quiera tener la información sobre sus empleados, usuarios de red, etc., catalogada y accesible desde multitud de aplicaciones. Además, el uso de un servicio de directorio facilita la gestión de la identidad de los usuarios de los sistemas de información en una organización.

En este módulo estudiaremos los conceptos básicos de los directorios, su diseño y su implantación. Nos centramos en LDAP (*lightweight directory access protocol*), dado que es el estándar más usado desde hace ya algunos años. Estudiaremos su concepto, utilidad, diseño e implantación.

Objetivos

Los objetivos que el estudiante habrá alcanzado al finalizar este módulo son:

1. Comprender el concepto y utilidad de un servicio de directorio.

2. Comprender el concepto de espacio de nombres.

3. Conocer las operaciones que ofrece un servicio de directorio como LDAP.

4. Diseñar un espacio de nombres para un servicio de directorio.

5. Diseñar el esquema de un servicio de directorio, manejando los conceptos de atributo y clase.

6. Comprender qué aspectos inciden en la seguridad, la eficiencia y la disponibilidad en un servicio de directorio.

7. Conocer implantaciones de servicios de directorio.

1. Concepto y uso de los directorios

La idea de directorio está relacionada con la información. Desde los inicios de los sistemas de ficheros, hasta llegar a los primeros sistemas operativos para ordenadores PC, la palabra *directorio* se ha asociado al esquema con el que los distintos archivos están organizados en las unidades de almacenamiento. En inglés, *telephone directory* es el nombre con el que se conoce a la guía telefónica: la relación de abonados al servicio telefónico, sus números de abonado y opcionalmente la dirección física de su domicilio. Sea como fuere, el concepto de directorio está estrechamente ligado a la organización de datos.

Un **directorio** es una estructura jerárquica que organiza y almacena datos acerca de elementos. Es un tipo concreto de base de datos.

Así pues, los ficheros, pese a estar realmente guardados en el soporte sin organización aparente, se organizan de forma lógica en directorios y subdirectorios. En un directorio de sistema informático en red, la información que se guarda tiene que ver con los recursos del sistema (servidores, impresoras, etc.) y con sus usuarios. Además, es preciso que el sistema cuente con un servidor para poder consultar la información, almacenarla o simplemente diseñarla. Esta información, como veremos más adelante, puede estar almacenada de forma distribuida y/o replicada. En consecuencia, es necesario contar con un servicio de directorio.

Un **servicio de directorio** es una plataforma que proporciona métodos para gestionar y almacenar los datos que contiene el directorio.

Un servicio de directorio permite la búsqueda de valores a partir de un determinado nombre (o identificador), de forma similar a lo que hace un diccionario. Así como un vocablo tiene distintas acepciones, hay distintos vocablos apuntando a un mismo significado, hay palabras derivadas, familias de palabras, etc., es decir, los objetos que almacena un directorio pueden estar relacionados de múltiples formas.

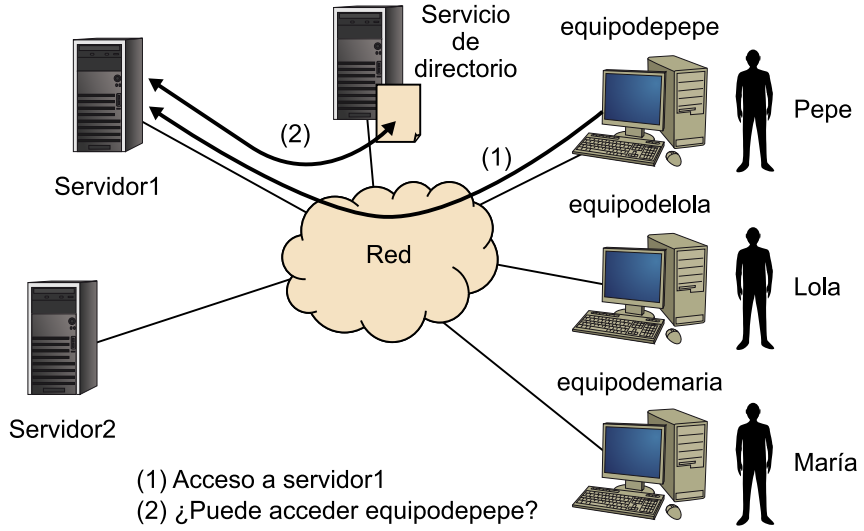
Ejemplo de un servicio de directorio

Supongamos un servicio que, dentro de un entorno de red local, devuelve una dirección IP y una serie de características a partir de un nombre de equipo. También podemos considerar que hay nombres que definen a grupos de equipos sobre los cuales aplicar determinados permisos. Pues bien, una aplicación que guarde y permita manejar esta información sería un servicio de directorio. Cualquier aplicación podría obtener información sobre los equipos y grupos, sólo seria necesario acceder al servicio de directorio.

En la figura siguiente, hay un servicio de directorio que controla los equipos y recursos de una red. Cuando *Servidor1* recibe una petición desde *equipodepepe*, el recurso solicita

al servicio de directorio si este equipo tiene permiso para acceder al recurso. Dentro del servicio de directorio, se ha creado un espacio de nombres, es decir, una concreción de cuál es el identificador de cada recurso de la red. Más adelante estudiaremos a fondo este concepto.

Ejemplo de consulta a un servicio de directorio para acceso a recursos



La idea de un servicio de directorio es tan reciente (o tan antiguo, según como se mire) como el inicio de los sistemas basados en red. En 1988, la ITU[1] y la ISO[2] se unieron para llevar a cabo el desarrollo de una serie de estándares sobre servicios de directorio, conocido como X.500, o *directory access protocol* (DAP). Este sistema era complejo y además implementaba múltiples herramientas que muchos clientes no acabaron usando nunca. Esto no propiciaba un amplio uso de X.500, con lo que se empezó a trabajar en una versión "ligera": el LDAP (*lightweigth directory access protocol*), cuyo primer esbozo se publicó en el RFC 1487. Esta primera versión, diseñada por personal de la Universidad de Michigan, se utilizaba como pasarela. LDAP conectaba clientes de directorio con servicios X.500. Dada la aceptación de LDAP, se trabajó en que este sistema fuera realmente el servicio de directorio y no una mera pasarela. Así pues, LDAP se convirtió en un estándar *de facto*, sobre todo porque supuso la base para distintos productos de servicio de directorio que gozan de gran popularidad. En el decurso de este módulo, haremos referencia en general a LDAP.

(1)ITU es la sigla de Internacional Telecommunications Union, en castellano Unión Internacional de Telecomunicaciones.

(2)ISO es la sigla de Internacional Organization for Standardization en castellano Organización Internacional para la Estandarización.

RFC

Los *request for comments* (RFC) son documentos estándar o propuestas de estándar usados por la Internet Engineering Task Force (IETF).

Aunque el concepto de directorio se relacione con datos, bien cierto es que hay varias diferencias entre un servicio de directorio y una base de datos:

- En los directorios se realizan muchas más lecturas de datos que escrituras.
- Los directorios pueden modificar más fácilmente el diseño de las "entidades" que albergan. En cambio en una base de datos cambiar el diseño de ésta *a posteriori* puede ser más complejo.
- Los datos de los directorios suelen estar distribuidos y replicados con mayor frecuencia que en bases de datos.

- Los directorios permiten, en general, consultas simples, y no consultas que requieran la fusión de datos provenientes de varias tablas (consultas *join* de las bases de datos).

Ved también

En el siguiente subapartado profundizaremos en algunos de estos aspectos a través de ejemplos.

1.1. Ejemplos y tipos de directorios

Antes de pasar a estudiar conceptos concretos sobre los servicios de directorio, veremos una panorámica de distintos tipos de directorios en función de su propósito o implantación.

Un tipo sencillo de directorios es el que está incluido en **aplicaciones de software**, como por ejemplo las libretas de direcciones. Una aplicación informática de correo electrónico puede incluir un directorio donde cada entrada es un "contacto", y entre la información almacenada se encuentra, claro está, la dirección de correo electrónico del contacto.

Un paso más allá sería que esta aplicación de libreta de direcciones funcionara como una aplicación informática independiente, o quizás fuera un elemento más del sistema operativo. En este caso, sería preciso establecer un **estándar de intercambio de información** para que los demás programas pudieran hacer uso de esta libreta de direcciones.

Un ejemplo de estos sistemas sería el LDIF (*LDAP, data interchange format*) el cual es utilizado como medio habitual de exportación de datos de libreta de direcciones a un fichero de texto imprimible.

Esta aplicación podría ser una **aplicación de red** ejecutándose en un servidor. De este modo, la información de los contactos estaría disponible para todos los equipos clientes que consultaran al servidor. En este caso, sería necesario establecer un protocolo de comunicaciones a nivel de aplicación para poder realizar distintas operaciones:

- Consultas sobre la información de un contacto.
- Mensaje de error en caso de que no se encontrara el contacto.
- Opcionalmente, un protocolo de identificación del usuario.
- Operaciones de alta, baja y modificación de contactos sólo ejecutables por un usuario con permisos de administrador, etc.

Los **directorios de sistemas operativos en red** almacenan datos de recursos de una red. Algunos ejemplos son el Active Directory de Microsoft, o el eDirectory de Novell. Así como una libreta de direcciones puede estar integrada en una aplicación, ejecutándose en un sistema, o funcionando en un servidor, está claro que será una aplicación concreta: guardar información acerca de contactos. En el caso de los directorios de sistemas operativos en red, su uso es más amplio.

Sistema de nombres para Internet

Otro ejemplo de directorio de propósito específico es el sistema de nombres para Internet, DNS[3]. El acceso a servicios basados en Internet se realiza mediante conexiones o envío de datagramas hacia una determinada dirección IP. El sistema DNS resuelve, a partir de un nombre, cuál es la dirección IP del recurso. La particularidad de este sistema es que la información se encuentra distribuida por toda la red Internet. Por ejemplo, en función del TLD[4] (es decir, si el nombre de dominio se corresponde a un .com, a un .es, etc.) los datos estarán en uno u otro servidor. Los datos correspondientes a los nombres locales, en general nombres que identifican servicios o máquinas dentro de un determinado dominio, se hallan en servidores de nombres autorizados. Finalmente, existe la replicación de datos, siendo uno de los ejemplos los servidores de nombre raíz, aquellos que contienen información acerca de dónde se hallan los servidores de TLD. Estos, a su vez, también están replicados por temas de eficiencia.

(3)DNS es la sigla de *domain name service*, en castellano, servicio de nombres de dominio.

(4)TLD es la sigla de *top level domain*.

Número IP

IP es una dirección de Internet o número único que identifica una máquina conectada a Internet, por ejemplo, 85.34.123.170.

Así como DNS se puede ver como un servicio de directorio un tanto específico, los hay de **propósito general**. Este es el caso del servicio LDAP. Aunque en ciertos casos su uso se remite a tener información sobre los usuarios de una serie de servicios en red (permitiendo, por ejemplo, el acceso a múltiples servicios mediante un único usuario y contraseña), LDAP permite definir soluciones para un amplio espectro de escenarios.

1.1.1. Directorios y su uso en seguridad y autenticación

Una aplicación interesante de los servicios de directorio es la seguridad. Por una parte, un servicio de directorio puede utilizarse, como se ha apuntado anteriormente, para gestionar la información de los usuarios de varios servicios de red. En concreto, se podría pensar en un servicio de directorio para una organización que albergue, para cada usuario de un sistema, la siguiente información:

- Nombre y apellidos
- Departamento de la organización
- Identificador de usuario
- Contraseña
- Fecha del último cambio de contraseña

En la organización se dispondría de varios servicios en red: un correo electrónico, un calendario, un sistema de compartición de documentos y una aplicación web de gestión de nóminas. Mediante un único identificador de usuario y su correspondiente contraseña, los usuarios de la organización podrían autenticarse ante cualquiera de los servicios. En realidad, el servidor de correo electrónico hace una consulta al servicio de directorio para comprobar que exista el identificador de usuario y que, evidentemente, la contraseña introducida sea la correcta.

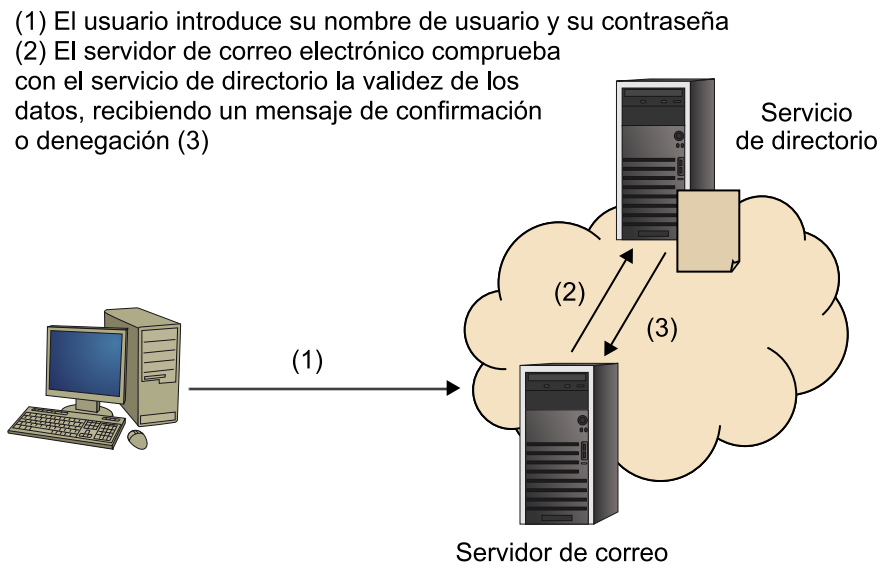
Si en cualquier momento el usuario desea cambiar su contraseña, debería realizar esta operación mediante el servicio de directorio. A su vez, este servicio podría avisar, por ejemplo, mediante un correo electrónico, de la próxima ca-

ducidad de la contraseña de un usuario. O bien si un usuario tiene la contraseña caducada, el propio servidor de correo electrónico mostraría un aviso de imposibilidad de autenticar al usuario debido a la caducidad de la contraseña.

Finalmente, en lo referente a la aplicación web de gestión de nóminas, cuando un usuario acceda, se consultará al servicio de directorio del departamento al cual pertenece. Si el usuario no pertenece al departamento, por ejemplo, de "gestión", sólo podrá consultar sus nóminas. En caso contrario, podrá realizar consultas sobre sus nóminas y las de los otros trabajadores, así como la creación y cálculo de nuevas nóminas.

En la figura siguiente se muestra este ejemplo de uso del servicio de directorio, en concreto, la autenticación al servidor de correo de la organización.

Uso del servicio de directorio como apoyo en la autenticación de un usuario ante un servicio de correo electrónico



Otro de los sistemas que se basan en un directorio es la infraestructura de clave pública (*public key infrastructure*, PKI). Estas plataformas realizan no sólo la distribución de certificados y claves a clientes y servidores, sino que además se encargan de otras funciones, siendo la revocación de claves y certificados una de las más importantes.

Los directorios ayudan a resolver dos de los problemas que suele acarrear la implantación de una PKI. El primero de ellos es la gestión de ciclo de vida de un certificado, es decir, cómo se crean, mantienen y destruyen los certificados. El segundo problema hace referencia a la localización de certificados. Es decir, cómo hallar confiablemente las claves públicas y certificados necesarios para comunicarse con servicios e individuos.

Usar un directorio puede claramente dar solución a estos problemas. El servicio de directorio actúa de punto central de administración durante el ciclo de vida de la PKI. La creación de certificados y claves se realiza a través del mismo

servicio de directorio. Cuando es preciso revocar un certificado, el directorio proporciona la herramienta adecuada: el directorio ofrece información sobre la lista de certificados revocados (y que ya no son válidos). Las propiedades de redundancia y distribución de la información son igualmente necesarias en entornos de PKI, siendo claramente proporcionadas por los servicios de directorio.

Una vez vistos distintos ejemplos de servicios de directorio y algunas de sus características, nos centraremos en la identificación de objetos y las operaciones básicas que utilizan los clientes del servicio de directorios.

1.2. Espacio de nombres

En los servicios de directorio, cada objeto está identificado mediante un nombre. Podríamos definir el espacio de nombres de un directorio como el conjunto de aquellos identificadores que se utilizan, o se pueden utilizar potencialmente, para identificar de forma unívoca los objetos del directorio.

> El identificador de objeto debe ser un nombre único dentro del servicio de directorio.

Además de identificar objetos, los identificadores también pueden identificar grupos de objetos, con lo cual se puede diseñar una estructura jerárquica. Veamos a continuación algunos ejemplos de espacios de nombres.

1.2.1. Sistema DNS

En un sistema DNS, el espacio de nombres permite identificar unívocamente a un equipo conectado a Internet. De hecho, varios nombres pueden apuntar a un mismo equipo. Para identificar un equipo dentro del espacio de nombres, el DNS se ayuda de una estructuración jerárquica iniciada en el dominio '.', que es el dominio raíz.

A partir de este dominio, se establecen una serie de dominios de nivel superior, los TLD apuntados anteriormente. Se establecen TLD geográficos (.es, .fr, .pt, .uk, etc.) y genéricos (.com, .net, etc.). Hay una serie de entidades acreditadas para gestionar los nombres pertenecientes a cada uno de los TLD. Por ejemplo, hay una entidad encargada de gestionar los dominios .cat. Cuando una organización quiere tener visibilidad pública a Internet, suele solicitar un dominio a la entidad correspondiente a su TLD (a no ser que para visitar sus servicios se deba usar una dirección IP, que al ser una serie de números, puede ser difícil de recordar).

Además, la organización puede tener la potestad de albergar un servidor de DNS dentro de sus sistemas y poder definir internamente los nombres de dominio para que sus servicios internos puedan ser accedidos desde el exterior. En este sentido, se pueden definir nombres para distintos niveles de dominio, hasta llegar a identificar a un servicio o equipo.

Ejemplo

Supongamos que una organización universitaria como la UOC solicita un nombre de dominio *uoc.edu*. Esta organización podrá decidir establecer una jerarquía de servicios Internet, a través de los distintos niveles del sistema de nombres. Por ejemplo, supongamos que quiere utilizar tres campus virtuales distintos en función de los estudios que albergue. Los dominios podrían ser los siguientes:

- Dominio de los estudios de Ingeniería: *ingenieria.uoc.edu*
- Dominio de los estudios de Ciencias: *ciencias.uoc.edu*
- Dominio de los estudios de Humanidades: *humanidades.uoc.edu*

Además, se podría pensar que en los estudios de informática hay un servicio web que alberga las páginas sobre docencia, otro que contiene las páginas correspondientes a grupos de investigación y sus publicaciones, y una última web con información de administración de estudios (expediente, matrículas, etc.). En este caso, se podrían definir los siguientes dominios:

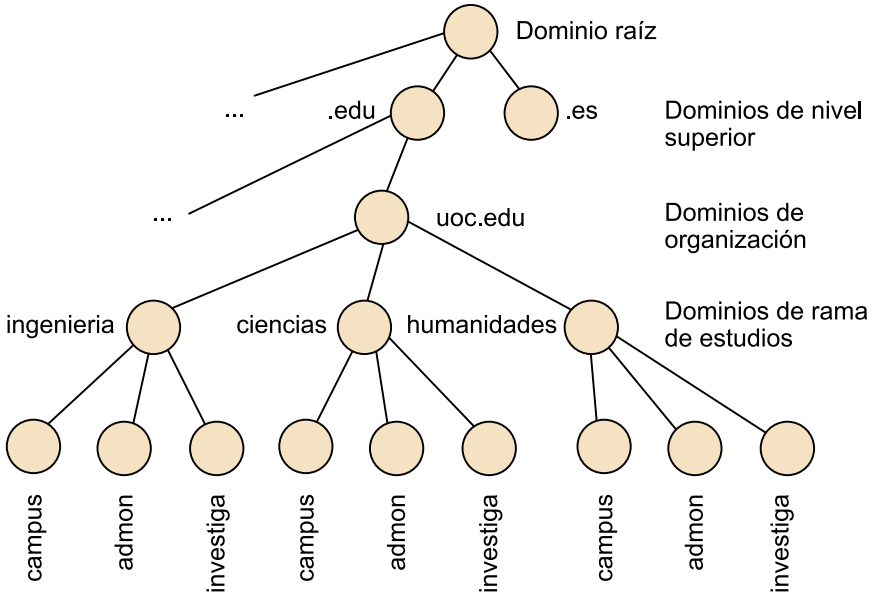
- *campus.ingenieria.uoc.edu*
- *investiga.ingenieria.uoc.edu*
- *admon.ingenieria.uoc.edu*

El resultado sería la identificación unívoca de distintos servicios, aun coincidiendo los nombres en alguno de los niveles:

- *campus.ingenieria.uoc.edu* es distinto de *campus.ciencias.uoc.edu*
- *campus.ingenieria.uoc.edu* es distinto de *investiga.ingenieria.uoc.edu*

La figura siguiente muestra la jerarquía definida mediante estos niveles de DNS.

Ejemplo de jerarquía de un sistema DNS



Aunque el DNS fue creado con el propósito de que los equipos sean accesibles desde Internet, cualquier equipo de una red sin dirección de red pública, o bien que los sistemas de conexión a la red no permitan su conexión desde el

exterior de ésta, puede disponer de un nombre que lo identifique dentro de un espacio de nombres. A través de este nombre se pueden establecer conexiones internas entre equipos de la misma red.

1.2.2. Sistema WINS

El sistema WINS (*Windows Internet name service*) es, tal y como su nombre indica, un equivalente a lo que ofrece DNS pero pensado para equipos Windows. Actualmente, los sistemas operativos de Microsoft permiten la convivencia de ambos servicios de nombres, de modo que es posible identificar un equipo a través de un nombre dentro de un espacio DNS y también un nombre WINS. Aplicaciones como la compartición de recursos mediante Windows se resolverán mediante el nombre WINS.

El espacio de nombres WINS fue diseñado para dar soporte al sistema NetBIOS[5]. Este sistema proporcionaba conectividad de red a mediados de los años 1980 a los ordenadores PC. Este sistema fue heredado y reimplementado por Microsoft, que lo llegó a incluir como estándar en sus primeras implementaciones de Windows permitiendo así el trabajo en red. Por consiguiente, WINS se considera un sistema de resolución de nombres para las aplicaciones que usan NetBIOS.

(5)NetBIOS significa Network Basic Input/Output System.

Estos nombres tienen una longitud de 15 caracteres ASCII para identificar el equipo. Los sistemas Windows pueden prescindir de este servicio de nombres, puesto que para resolver los nombres NetBIOS en pequeñas redes, basta con hacer peticiones *broadcast*. Es decir, que se difunden por toda la red. Además, para evitar la generación de este tráfico de red, se puede usar un fichero (lmhosts) como servicio de directorios para la resolución de nombres.

A pesar de que NetBIOS y WINS gozaron de gran popularidad en los sistemas operativos y redes basadas en productos Microsoft, el auge de Internet hizo abandonar estos sistemas y pasar a especificar nombres en el sistema DNS.

1.2.3. Sistema LDAP

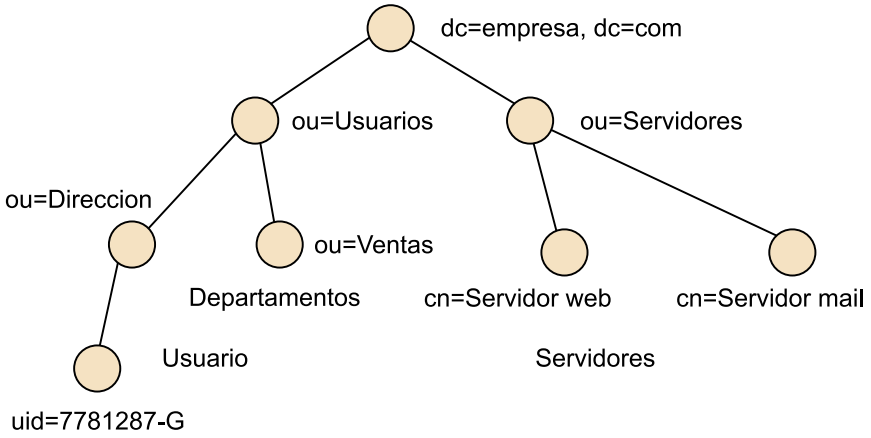
El sistema LDAP, con sus distintas versiones, se ha convertido en un estándar genérico de servicio de directorio. Tanto es así que muchos de los productos ampliamente usados para implantar servicios de directorio se basan en LDAP. Por este motivo, ahora nos centraremos en cómo es el espacio de nombres que se define en LDAP. Este sistema hereda muchas características de los sistemas X.500.

El sistema de nombres que usa LDAP permite la organización de los objetos de forma jerárquica.

Ejemplo de directorio LDAP

En la figura siguiente se puede observar un ejemplo de directorio LDAP.

Ejemplo de espacio de nombres de un directorio basado en LDAP



En esta organización (*empresa.com*) hay definidos dos grupos de objetos: los usuarios y los servidores. Los usuarios se dividen en departamentos. Como veremos más adelante, LDAP y otras implementaciones derivadas permiten la agrupación de elementos y crear una jerarquía. LDAP no fija ninguna jerarquía ni ningún número determinado de niveles: el espacio de nombres permite dar flexibilidad para adaptarse a multitud de usuarios.

Cada objeto se plasma como una **entrada de directorio**. En el ejemplo hay un total de ocho entradas. Cada entrada tiene un nombre *distinguished name* (DN). Por ejemplo, la organización tiene como DN `dc=empresa, dc=com`.

Cada entrada de directorio está compuesta por una serie de **atributos**, cada uno de ellos describe varios aspectos del objeto que la entrada identifica. En el ejemplo se define un usuario. Este objeto podría tener los atributos descritos en la tabla siguiente.

Ejemplo de una entrada LDAP

Atributo	Valor
`objectclass`	`person`
`cn`	`José María López García` `Pepe López García`
`sn`	`López García`
`telephoneNumber`	`1789`
`mail`	`josem.lopez@empresa.com`
`jpegPhoto`	`nU6KNyVIYS817zVdf5YKF1FrNb...`

La definición de qué atributos forman parte del directorio se conoce como el **esquema del directorio**. A continuación se explica el significado de los anteriores atributos:

- **`objectclass`**. Especifica a qué clase pertenece el objeto.
- **`Common name (cn)`**. Nombre del usuario, puede tener más de un valor. En este caso, se considera nombre del usuario tanto José María, como Pepe.

Lectura recomendada

El RFC 2256 da más detalles sobre los atributos.

Ved también

En el apartado 2 de este módulo profundizaremos en el concepto de esquema y su diseño a través de reglas.

Ved también

En el apartado 2 sobre diseño del directorio profundizaremos en el concepto de objeto.

- **`Surname (sn)`**. Es el apellido del usuario. Puede ser útil para poder ordenar alfabéticamente por apellido.

- **`telephoneNumber`**. Como su nombre indica, sirve para almacenar el número de teléfono del usuario. En este caso, se trata de una extensión de centralita.

- **`mail`**. Almacena la dirección de correo electrónico.

- **`jpegPhoto`**. Contiene una pequeña imagen del usuario.

En el ejemplo de la figura anterior, se utilizan además los atributos **`domain component (dc)`** y **`organizationa unit (ou)`**.

En LDAP cada objeto se identifica mediante un nombre, el ***distinguished name*** **(DN)**, formado por varios atributos y sus valores.

En el ejemplo de la figura anterior, podríamos decir que hay un usuario con DN `uid=7781287-G, ou=Direccion, ou=Usuarios, dc=empresa, dc=com`. Un atributo concreto, por ejemplo `uid=77781287-G` se conoce como ***relative distinguished name*** **(RDN)**.

Por norma, el valor de RDN para cada nivel debe ser único. Por lo tanto, no podría haber dos usuarios con RDN `uid=7781287-G` situados en el grupo "Direccion". Lo que sí estaría permitido es la existencia de un usuario, también con RDN `uid=7781287-G` pero que perteneciera al grupo de "Ventas".

Para construir el DN se usa toda la cadena de RDN desde la hoja del árbol hasta la raíz. Una variante del RDN se da cuando, en lugar de usar un único atributo como RDN, se usan varios atributos de la entrada. Así pues, es posible tener dos entradas con el RDN `cn=Servidor web`, siempre y cuando se use otro atributo, por ejemplo, `description`, para diferenciar entre las dos entradas. En este caso, se trata de un RDN multivalor (en inglés *multivalued*), de modo que el objeto DN `cn=Servidor web+description=Web tienda, ou=Servidores, dc=empresa, dc=com` es distinto al objeto DN `cn=Servidor web+description=Web nominas, ou=Servidores, dc=empresa, dc=com`. Aunque sea posible el uso de varios atributos de una entrada de directorio a fin de identificar una entrada, se recomienda que sólo se use un atributo de cada entrada, por motivos básicamente de eficiencia.

Identificación mediante DN

En cierto modo, la identificación mediante DN recuerda la identificación de un fichero dentro de una estructura de directorios (por ejemplo, /usr/bin/grep).

Ved también

En el apartado 2 profundizaremos en el diseño del espacio de nombres de un servicio de directorio basándonos en LDAP.

1.3. Operaciones de cliente

Los directorios no tendrían sentido sin herramientas que permitieran su consulta. Por una parte, existen herramientas que pueden conectar con el servicio de directorio para poder hacer consultas sobre los usuarios y recursos. Por otra

parte, hay servicios que consultan al directorio con distintos fines, como por ejemplo, validar un usuario o comprobar los permisos de acceso a un servicio. Además es necesario disponer de herramientas para modificar la información que contiene el directorio. En este subapartado definimos las operaciones más frecuentes a nivel de interacción de un cliente con un servicio de directorio.

1.3.1. Operaciones de interrogación

Tal y como su nombre indica, este tipo de operaciones permiten al cliente buscar información. En concreto, se definen dos operaciones básicas: la comparación y la búsqueda. La comparación se usa para comprobar si una entrada en particular contiene un valor concreto para un atributo. El servidor responderá cierto o falso en función del resultado de esta comparación. La comparación tiene un uso en casos muy limitados. En cambio la operación de búsqueda es mucho más potente y permite, en el fondo, realizar también tareas de comparación. La operación de búsqueda es una herramienta potente de interrogación a servidores de directorio.

La operación de **búsqueda** (*search*) permite buscar en el directorio y obtener información de las entradas.

Esta operación dispone de hasta ocho parámetros, que se describen a continuación:

1) ***Base***. A partir de qué entrada u objeto se quiere empezar a hacer la búsqueda.

2) ***Scope***. Permite definir el ámbito de la búsqueda, los valores posibles son:

a) ***Onelevel***: permite buscar sólo en el nivel siguiente al definido en el parámetro base;

b) ***Sub***: se utiliza para buscar en todo el subárbol, es decir, desde la base hasta las hojas;

c) ***Base***: se usa para buscar sólo en la propia base, con el objetivo de obtener información de la misma.

3) ***Alias deferencing options***. LDAP permite la definición de alias, mediante los cuales se pueden enlazar entradas del directorio (una es el alias y la otra el objeto original).

Nota

La operación de búsqueda permite tratar con alias, pero este concepto esta fuera del objetivo de este módulo.

4) ***Size limit***. Este parámetro indica al servidor el número máximo de entradas halladas con éxito que se quiere obtener.

5) ***Time limit***. Especifica el tiempo máximo en segundos durante el cual se puede ejecutar la búsqueda. Si el valor es 0 significa que no hay tiempo límite establecido.

6) ***Attributes-only***. Permite especificar al servidor de directorio que devuelva únicamente cuáles son los atributos que contienen las entradas encontradas, y no sus valores.

7) ***List of attributes to return***. Es la lista de los atributos que se quiere obtener. Si no se especifican, se entiende que se quieren obtener todos los atributos.

8) ***Search filter***. El último de los parámetros que se definen para la operación de búsqueda de LDAP es el *search filter*. Es una expresión que describe el tipo de entradas que se quiere obtener. Sin pretender ser exhaustivos, a continuación mostramos algunos ejemplos de filtros de búsqueda, juntamente con su significado:

Lectura recomendada

Para saber más sobre el *search filter* podéis consultar la obra siguiente:

T. A. Howes y otros (2003). *Understanding and Deploying LDAP Directory Services* (2.ª ed.). Addison-Wesley.

- **(sn=prados)**. Este criterio de búsqueda significa que devuelva las entradas con el atributo "sn" (apellido) con el valor "prados".

Lectura recomendada

El RFC2254 recoge la definición de los filtros de búsqueda de LDAP, y ejemplos para su mejor comprensión.

- **(sn=mar*)**. En este ejemplo el asterisco indica que el criterio es que el apellido empiece por *mar*, es decir: marquez, martin, martinez, etc. El asterisco permitiría buscar los apellidos que terminen con *ez*, **(sn=*ez)**.

- **(sn~=fernandez)**. En este caso, se retornarían apellidos parecidos a fernandez (hernandez, ferrandiz, etc.).

- **(age>=18)**. Este filtro retornaría los usuarios mayores de edad. Se debe tener en cuenta que (age<18) no es posible. Siempre debe haber un igual en la comparación para los filtros LDAP. En este caso, podríamos usar una negación, **(!(age>=18))**. Los operadores ">=" y "<=" se pueden usar también en cadenas de caracteres (por ejemplo, apellidos), en cuyo caso se usará un orden lexicográfico.

- **(jpegPhoto=*)**. Devuelve todas las entradas que incluyen una imagen JPEG.

JXplorer

JXplorer es una herramienta de software libre para conectarse a servidores LDAP y obtener información. Dispone de una interfaz gráfica de usuario.

También pueden construirse filtros compuestos mediante los operadores lógicos "&" (and) y "|" (or).

Por ejemplo, con la expresión **(&(objectClass=person)(|(givenName=Pedro)(!(age<=50))))** se obtienen las entradas de clase persona cuyo nombre de pila sea Pedro y sea mayor de 50 años.

ldapmodify

Así como ldapsearch es una herramienta para consultar información, ldapmodify permite la variación de los datos contenidos en el directorio.

Las implementaciones de cliente LDAP nos ofrecen una herramienta de búsqueda. La herramienta puede tener una interfaz gráfica de usuario, en cuyo caso dispondremos de cuadros de diálogo para introducir los anteriores pará-

metros de búsqueda, o bien dispondremos de asistentes que nos ayudarán a su creación. La herramienta puede ser mediante la línea de comandos, como **ldapsearch**.

Uso de ldapsearch

Un ejemplo de uso de la herramienta sería ldapsearch:

```
ldapsearch –h ldap.ejemplo.com –s sub –b "ou=ingenieros" "(cn~=Juan Prados)"
```

En este caso se busca en el servidor ldap.ejemplo.com, dentro del apartado de ingenieros, la entrada correspondiente a un tal Juan Prados. La herramienta ldapsearch ha realizado una consulta al servicio de directorio que no ha precisado de una conexión autenticada o, lo que es lo mismo, ha usado una **conexión anónima**.

1.3.2. Operaciones de actualización

LDAP dispone de cuatro operaciones de actualización de datos: añadir, borrar, modificar y renombrar/mover.

La operación de añadir (**add**) permite crear una nueva entrada e introducir sus datos. Como parámetros de la operación, está el DN para la nueva entrada (que servirá, evidentemente, para situar la entrada dentro del árbol) y la lista de tributos con sus valores. Esta lista de valores debe coincidir con el esquema del directorio, es decir, con el modelo de datos en cuanto a atributos se refiere, que estudiaremos más adelante.

La operación de borrar (**delete**) elimina una entrada del directorio. Solamente precisa del DN de la entrada que se quiere eliminar. Es importante que la entrada no tenga hijos en el árbol para poder eliminarla.

La operación de modificar (**modify**) permite cambiar valores de atributos de una entrada, añadirlos o bien borrarlos.

Finalmente, una operación más compleja a la vez que potente es la de renombrar. Esta operación (**rename**) permite cambiar el DN de las entradas. Así pues, en el fondo, permite también mover la entrada de una ubicación a otra del árbol. Tiene cuatro parámetros:

1) El DN de la entrada a renombrar/mover.

2) El nuevo RDN que tendrá la entrada.

3) El DN del que será el nuevo padre de la entrada (el parámetro es opcional si se emplea usando un DN distinto al del padre actual de la entrada, se realizará un movimiento de la entrada).

4) Finalmente, un indicador para borrar el antiguo RDN. Si no se borra el antiguo RDN, el nuevo RDN de la entrada se añade como atributo de la entrada.

1.3.3. Operaciones de autenticación

La conexión a un directorio no está exenta de las implicaciones en la seguridad del propio servicio de directorio. Por ejemplo, puede ser que la consulta de información sea pública para cualquier usuario, mientras que la modificación sea sólo posible para ciertos usuarios con rol de administradores del servicio de directorio. Por otra parte, si la implementación del directorio guarda datos sensibles, como por ejemplo contraseñas, sin cifrar (nada recomendable), se deberán tomar precauciones especiales para garantizar la seguridad de los datos. Para realizar una conexión (operación **bind**), se especifica el DN de quien realiza la conexión. Se puede usar una contraseña para autenticación, así como distintos métodos de seguridad. Estos modelos se revisarán más adelante en el subapartado 2.3.

1.3.4. Acceso desde otras aplicaciones

Otra forma en la que se encuentra disponible LDAP es como interfaz para implementar programas (API). De este modo, por ejemplo, en lenguaje C es posible usar funciones contra un servicio de directorio LDAP, como `ldap_search()`, `ldap_bind()`, `ldap_add()`, etc.

También existen interfaces para poder acceder a servicios LDAP desde C#, C ++ o PHP.

En lenguaje Java, el *Java naming and directory interface* (JNDI) proporciona herramientas para usar servicios de directorio.

El siguiente ejemplo muestra una conexión anónima a un servicio LDAP mediante C#.

```
LdapConnection ldapConn= new LdapConnection();

ldapConn.Connect ("ldap.ejemplo.com",389);

ldapConn.Bind (null, null);
```

Y este ejemplo hace lo mismo, pero esta vez en PHP.

```
<?php

$ldapconn = ldap_connect("ldap.ejemplo.com")
    or die("Imposible conectar.");


if ($ldapconn) {
    $ldapbind = ldap_bind($ldapconn);
```

```
}

?>
```

Finalmente, este ejemplo ilustra la conexión a un directorio LDAP mediante JNDI.

```
DirContext ctx = new InitialDirContext(env);

env.put(Context.INITIAL_CONTEXT_FACTORY,"com.sun.jndi.ldap.LdapCtxFactory");

env.put(Context.PROVIDER_URL, "ldap://direccion:389");

env.put(Context.SECURITY_AUTHENTICATION,"simple");

env.put(Context.SECURITY_PRINCIPAL,"cn=usuario");

env.put(Context.SECURITY_CREDENTIALS,"contrasena");
```

2. Diseño del directorio

El espacio de nombres es un elemento estrechamente ligado con la organización de los objetos que incluye el directorio. Tal y como hemos visto, en LDAP los objetos se identifican mediante una serie de valores específicos de atributos, en principio uno para cada nivel del espacio de nombres.

Así pues, se puede pensar que debería haber cierta relación entre la organización de la institución y el diseño de su directorio. Por otra parte, será necesario tener claro qué elementos conviene almacenar para cada entrada, es decir, cuáles serán los atributos que definirán los objetos del directorio. En este apartado, hacemos énfasis en los aspectos de diseño del directorio, tanto a nivel de espacio de nombres como a nivel del esquema. También realizamos algunas reflexiones en torno a la seguridad, robustez y eficiencia del sistema.

2.1. Diseño del espacio de nombres

Para el diseño del espacio de nombres se pueden tener en cuenta tres elementos: la elección del sufijo, la estructura o complejidad del directorio y la identificación de objetos dentro del directorio.

2.1.1. Elección del sufijo

La organización para la cual se diseña el directorio puede tener un ámbito local o bien puede formar parte de una organización mayor, que también cuente con un directorio. Sea como fuere, lo importante es que el sufijo (el nombre de la parte "raíz" del árbol del directorio) identifique unívocamente la organización. Hay tres alternativas válidas para elegir el sufijo:

1) La primera de ellas consiste en cumplir la recomendación de la RFC 2247. En ella se especifica que es conveniente mapear en DN del directorio con el nombre DNS que la organización tenga asignado (o pretenda tener asignado algún día).

> Si la organización tiene por ejemplo una página web con el dominio www.empresa-ejemplar.com, sería conveniente que el DN del dominio, el sufijo, fuera DN `dc=empresa-ejemplar, dc=com`.

2) Una segunda alternativa, ligeramente distinta a la primera, consiste en usar todo el nombre de dominio, esta vez usando el atributo "o" (de **organization**). En este caso DN `o=empresa-ejemplar.com`.

3) Finalmente, la tercera alternativa tiene que ver con la localización geográfica de la empresa, tal y como se formulaba en las recomendaciones para X.500.

Estas nomenclaturas se usan, por ejemplo, para identificar organismos y entidades dentro de los certificados electrónicos. Para ello, se usa el atributo "c" (**country**). En el caso de que la empresa esté ubicada en España y se llame Empresa Ejemplar, S. A., tendríamos DN `o=Empresa Ejemplar\, S. A., c=ES`. Nótese el uso del carácter '\' para denotar que la coma pertenece al valor del atributo y el intérprete no debe confundirlo con el inicio de un nuevo atributo.

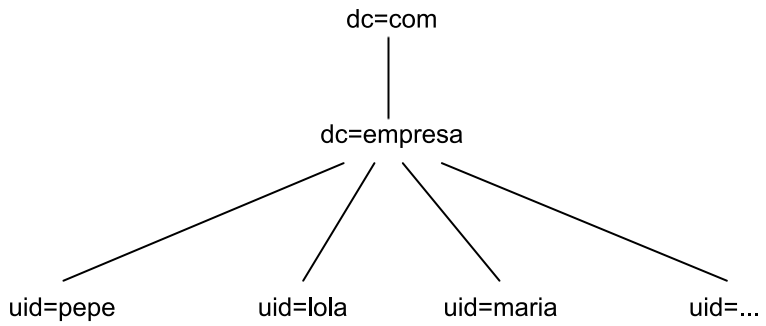
Finalmente, es posible que un directorio tenga más de un sufijo. Esto será necesario en caso de fusionar directorios pertenecientes a organizaciones con sufijos distintos.

2.1.2. Estructura del directorio

Otro elemento a tener en cuenta al diseñar un espacio de nombres es la estructura del directorio, que en el fondo también tendrá implicaciones en la complejidad del directorio en cuanto a jerarquía se refiere.

El espacio de nombres más simple sería un espacio de nombres plano, por ejemplo, sin departamentos ni grupos de usuarios. La figura siguiente muestra un espacio de nombres plano.

Espacio de nombres plano



Esto sería adecuado para organizaciones con pocos usuarios y/o recursos. Ahora bien, en grandes organizaciones es recomendable seguir una estructura jerárquica.

En organizaciones que contemplan departamentos o distintos perfiles de usuarios, se aconseja crear unidades organizacionales (identificadas con el atributo "ou"). Además, distribuir los recursos en grupos puede ser útil, por ejemplo, para temas de control de acceso a recursos.

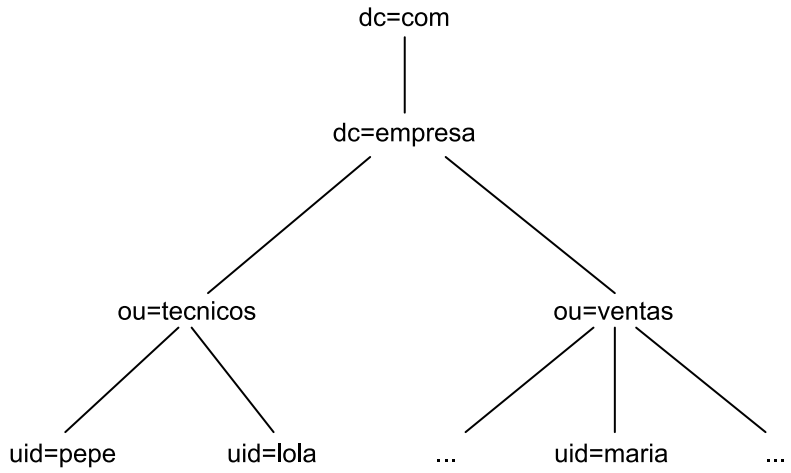
Ved también

El control de acceso a recursos se trata en el módulo "Identificación, autenticación y control de acceso".

Ejemplo

Supongamos la distribución de usuarios de una empresa como la que muestra la figura siguiente.

Ejemplo de estructura jerárquica



Usando una estructura como la anterior, sería fácil permitir el acceso a determinados recursos a los usuarios del grupo "Técnicos" o a los del grupo "Ventas".

2.1.3. Identificación de objetos en el directorio

El último elemento que analizaremos sobre el diseño del espacio de nombres tiene que ver con la identificación de los objetos dentro del directorio. Tal y como hemos visto, las restricciones que impone LDAP a la hora de identificar objetos son dos:

- El RDN de una entrada se formará a partir de un atributo (o más de uno, aunque no es recomendable).
- En RDN debe ser único entre las entradas "hermanas" (aquellas que penden del mismo padre en la estructura).

Aunque solamente haya establecidas estas dos restricciones, se aconseja en general que la identificación de los objetos (en especial cuando se trata de personas o usuarios) se haga a través de un identificador único.

Para hacer esto, se debe garantizar que el nombre de usuario sea único. Claramente, podría haber dos personas con el mismo nombre. Aunque una organización sea pequeña, resultaría poco práctico que hubiera un usuario identificado con un nombre muy común, como por ejemplo "pepe", porque fácilmente puede haber o habrá algún día otro usuario "pepe" que almacenar en el directorio (y no sería muy estético llamarlo "pepe2"). Por este motivo debe utilizarse un atributo que identifique unívocamente al usuario, por ejemplo se puede usar su NIF.

Claramente, otra opción para gestionar la unicidad de los atributos de identificación es que el administrador cree los identificadores controlando que estos sean únicos. Podrían usarse las iniciales, y adicionalmente números, para evitar repeticiones. O bien si el directorio se usa para acceder a servicios funcio-

nando en sistemas operativos tipo Unix o Linux, estaría bien que el identificador de usuario coincidiera con el nombre de usuario que se defina para los servicios. En este caso se utilizaría el atributo "uid", que significa *user identifier*.

Finalmente, otra opción, aunque no recomendable, sería asignar una cadena aleatoria como identificador del usuario y decimos que no es recomendable porque recordar esta cadena aleatoria no sería fácil.

Pese a tener múltiples opciones para identificar a usuarios y recursos, la solución debe ser práctica y estética.

2.2. Esquema del directorio

Hasta ahora hemos hablado de objetos en el sentido amplio de la palabra. Hemos visto que un directorio contiene entradas con sus distintos atributos. Una entrada representa a un objeto cuya información es almacenada en el directorio. Y se puede entrever que las entradas pueden ser de varios tipos: individuos, recursos, grupos, etc. Anteriormente ya hemos apuntado en qué consiste el esquema del directorio, ahora vamos a tratar varios aspectos relacionados con su diseño.

El **esquema del directorio** es la definición de qué tipos de objeto guarda un directorio y qué atributos se utilizan para su definición.

Las implementaciones de servicios de directorio ya suelen incluir sus propias definiciones de esquema, los cuales suelen poderse ampliar sin problemas para cubrir cualquier necesidad.

En el decurso de este subapartado vamos a tratar los conceptos de atributo y clase, esenciales en la definición del esquema del directorio.

2.2.1. Atributo

Los atributos sirven directamente para guardar información (nombre de persona, número de teléfono, fotografía, etc.). Para definir un atributo en LDAP, es preciso contar con una serie de información:

- Un nombre que identifica al atributo que se define. En caso de LDAP, ya hay algunos nombres estándar definidos (*common name*, *telephoneNumber*, etc.), algunos de ellos con una abreviatura también conocida (por ejemplo, "cn" para *common name*). LDAP no distingue entre mayúsculas y minúsculas.

- Un OID (identificador de objeto) que también identifique al atributo. Los OID son cadenas de números que permiten localizar de forma precisa un objeto de datos. Los OID también definen un espacio de nombres con una jerarquía. Por ejemplo, 2.5.4.16. es el OID de *postalAddress* y el valor "2" significa que el resto de estructura ha sido propuesto con la ITU en conjunción con la ISO. La notación que siguen los OID está definida en el ASN.1 (*abstract syntax notation one*). Dado que en el fondo usar el OID no es necesario a no ser que se deba interactuar con sistemas X.500, el uso de OID no suele ser muy popular, ya que es más cómodo trabajar con nombres a la hora de identificar el atributo.

- Una descripción textual del atributo para permitir anotaciones. Esta permite hasta 1.024 caracteres.

- Una sintaxis del atributo, la cual define cómo se representan los datos. La sintaxis se identifica mediante un OID.

  Por ejemplo, si se especifica 1.3.6.1.4.1.1466.115.121.1.15, nos estamos refiriendo a que la sintaxis del atributo es la cadena de texto. O bien, si se trata de un número de teléfono, empezando por '+', seguido del prefijo internacional y seguido del número de abonado, estamos refiriéndonos a la sintaxis 1.3.6.1.4.1.1466.115.121.1.50.

- Unas reglas de coincidencia, que son utilizadas en las búsquedas de información en el directorio. Por ejemplo, si se especifica *caseIgnoreMatch*, se está indicando que las mayúsculas no se deben tener en cuenta a la hora de comparar cadenas de caracteres.

- Otros valores, como por ejemplo la longitud máxima, etc.

Los atributos pueden derivar de otros atributos. Es decir, dada la definición general de un atributo como por ejemplo "name", puede haber una serie de atributos que deriven de éste y, en consecuencia, hereden sus características.

ASN.1

El ASN.1 define un marco de identificación de datos con independencia del método de representación que use una máquina. Corresponde a la capa de presentación de la OSI (*open systems interconnection*).

X.500

Recordad que X.500 fue el primer sistema estándar de directorio.

Enlace de interés

Podéis encontrar información acerca de los OID en la siguiente página web: http://www.oid-info.com.

Definición de nuevos atributos

Antes de definir nuevos atributos, conviene saber cuáles están definidos en la herramienta de servicio de directorio que usaremos. Para este objetivo, consultaremos los ficheros de definición de esquema que incluya nuestro producto de servicio de directorio.

Ejemplo

El esquema de datos de LDAP, que especifica los atributos estándar que ya vienen definidos para un servicio de directorio, incluye la definición del atributo "name":

```
attributetype ( 2.5.4.41 NAME 'name'

        EQUALITY caseIgnoreMatch

        SUBSTR caseIgnoreSubstringsMatch

        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
```

pero a la hora de definir el atributo "cn" o "commonName", especifica que su superior es "name":

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

En el ejemplo de la definición del atributo "name" se observa el uso de `EQUALITY` y `SUBSRT` para especificar reglas de coincidencia, mientras que la sintaxis se especifica mediante `SYNTAX` y su correspondiente OID. En la propia sintaxis se especifica una longitud máxima entre llaves.

Para finalizar con el subapartado sobre atributos, se presenta la siguiente tabla que contiene algunos de los atributos estándar en LDAP.

Descripción de algunos de los atributos más habituales en LDAP

Atributo	Descripción
cn, commonName	Nombre del objeto. Si el objeto es una persona, sirve para especificar su nombre completo
sn, surname	Apellido de una persona
serialNumber	Número de serie de un dispositivo o recurso
c, countryName	Nombre del país usando dos caracteres, tal y como especifica la ISO 3166
st, stateOrProvinceName	Nombre del estado, provincia, comunidad autónoma, etc.
street, streetAddress	Dirección postal (calle, número, etc.)
o, organizationName	Nombre de la organización
ou, organizationalUnitName	Nombre del departamento u otra unidad organizacional
title	Título de la persona dentro de una organización (presidente, director, etc.)
description	Descripción del objeto, de forma comprensible para los humanos
postalCode	Código postal
telephoneNumber	Número de teléfono
preferredDeliveryMethod	Descripción de cómo quiere una persona que se le entregue información (por ejemplo, por fax o e-mail)
member	Se trata de un DN que indica de quién es miembro el objeto en el árbol del directorio
uid, userid	Identificador de usuario, en general usado para autenticarse en un servicio o sistema

Atributo	Descripción
userPassword	Contraseña
dc, domainComponent	Especificación de un dominio DNS según RFCs 1274 y 2247, por ejemplo "es" o "uk"
buildingName	Nombre del edificio donde está ubicada una organización o una persona
preferredLanguage	Idioma preferido por la persona para comunicarse oralmente o por escrito
userSMIMECertificate	Certificado o cadena de certificados para autenticación y comprobación de firmas electrónicas

2.2.2. Clase de objeto

Hasta ahora hemos visto la definición de atributos. Mediante esta operación se pueden definir atributos específicos para nuestro servicio de directorio. Una vez los atributos han sido definidos, se pueden utilizar clases de objetos para definir cómo son las entradas del directorio. Cada entrada del directorio pertenece a un tipo de objeto determinado. Como en el caso de los atributos, ya hay algunas clases de objetos definidas como estándar.

Mediante las clases de objetos se especifica qué atributos debe contener la entrada de forma obligatoria y qué atributos puede contener definidos de manera opcional. También será posible usar la clase de objeto a la hora de buscar información en el directorio.

Ejemplo

Recordemos que en el ejemplo sobre operaciones de interrogación visto anteriormente, especificábamos *objectClass=person* para buscar sólo personas (en el fondo, entradas correspondientes a la clase "person"). Este es un ejemplo de definición de la clase "person":

```
objectClasses: ( 2.5.6.6 NAME 'person'

 DESC 'RFC2256: a person' SUP top STRUCTURAL

 MUST ( sn $ cn )

 MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )
```

Nótese que, de un modo similar a la definición de atributos en el esquema, la definición de una clase precisa del uso de un OID (2.5.6.6). También hay un campo de descripción textual donde se informa que la definición de "person" se corresponde con lo establecido en el RFC 2256.

Se especifica que la clase es de tipo estructural. Esto significa que la clase describe propiedades básicas del objeto; en general, casi todas las clases definidas son de este tipo. La descripción de otros tipos se escapa de los objetivos de este módulo.

El apartado `MUST` indica que es necesario que un objeto del tipo "person" incluya el apellido ("sn") y el nombre completo ("cn"). Los campos que puede contener, pero no son esenciales, están indicados en `MAY`. En la definición de algunas clases, no existe la entrada `MUST`. Esto significa que no se requiere ningún campo en concreto.

Finalmente, vemos que en las clases también existe la definición de "superiores", en este caso el superior de "person" es "top".

En la tabla siguiente se especifican algunas características para algunas de las clases típicamente definidas en un esquema.

Características de algunas de la clases típicamente definidas en un esquema

Clase	MUST	MAY
alias (permite que la entrada de directorio apunte hacia otra entrada)	aliasedObjectName	
organization	o	userPassword, seeAlso, businessCategory, preferredDeliveryMethod, telephoneNumber, facsimileTelephoneNumber, street, postOfficeBox, postalCode, postalAddress, physicalDeliveryOfficeName, description, etc.
organizationalUnit	ou	userPassword, seeAlso, businessCategory, preferredDeliveryMethod, telephoneNumber, internationaliSDNNumber, facsimileTelephoneNumber, street, postOfficeBox, postalCode, description, etc.
person	sn, cn	userPassword, telephoneNumber, seeAlso, description
device	cn	serialNumber, seeAlso, owner, ou, o, l (localización), description
account	userid	description, seeAlso, l, o, ou, host
document	documentIdentifier	commonName, description, seeAlso, l, o, ou, documentTitle, documentVersion, documentAuthor, documentLocation, documentPublisher
room	commonName	roomNumber, description, seeAlso, telephoneNumber

Nótese que clases como *organizationUnit* pueden tener contraseña o número de teléfono.

2.3. Seguridad, eficiencia y disponibilidad del directorio

Después de haber analizado los puntos clave del diseño del espacio de nombres y del esquema del directorio, veremos brevemente otros aspectos estrechamente relacionados con la implantación real del servicio de directorio.

2.3.1. Seguridad en el directorio

El objetivo de aplicar seguridad al directorio es proteger la información de accesos no autorizados. Este modelo va más allá de permitir el acceso al servicio de directorio, y puede incluso detallar qué operaciones se pueden llevar a cabo con qué atributos, y por parte de quién se pueden realizar estas operaciones.

Una opción para proteger la información es usar protección en la capa de transporte de la información. Así, por ejemplo, puede usarse un **canal de comunicación SSL/TLS**[6] para proporcionar confidencialidad y autenticidad a la información que se transmite entre el servicio de directorio y la herramienta con quien interactúa.

[6]SSL es la sigla de *secure socket layer*; TLS, de *transport layer security*.

Una de las informaciones que conviene proteger del acceso indebido es la contraseña. En los estándares no se especifica cómo debe hacerse esta protección, con lo cual se deja en manos del implementador. Lo que sí es habitual es que en lugar de guardar la contraseña en claro se guarde un resumen de la contraseña en lugar de la contraseña original.

Ved también

El almacenamiento de contraseñas se estudia en el módulo "Identificación, autenticación y control de acceso".

En el caso de la contraseña, guardando su resumen no es computacionalmente posible saber cuál es el valor original de ésta. Ahora bien, dado que la función de resumen que se utiliza para guardar la contraseña es conocida, sería relativamente fácil que un atacante creara una contraseña y guardara su resumen en la entrada de cualquier usuario. Mediante este ataque, el atacante podría suplantar la identidad del usuario ante cualquier servicio o sistema que usara el directorio como herramienta de autenticación.

Con relación a este tema, puede ser necesaria la definición de **listas de control de acceso**. Mediante estas listas el servicio de directorio tiene claramente definidos varios parámetros sobre el acceso a los datos, cuyo seguimiento debe procurarse ante el acceso de consultas y modificaciones.

Ejemplo

Para cada atributo de cada tipo de entidad, pueden definirse permisos de lectura y escritura en función del perfil de quien ejecuta la petición, además de poder especificar métodos de validación específicos. Así pues, para ver la extensión telefónica de un trabajador de una organización, no haría falta autenticarse (bastaría con una conexión anónima). O bien, para modificar el valor de un atributo "salario", debería ser necesario autenticarse como un usuario del grupo de administradores.

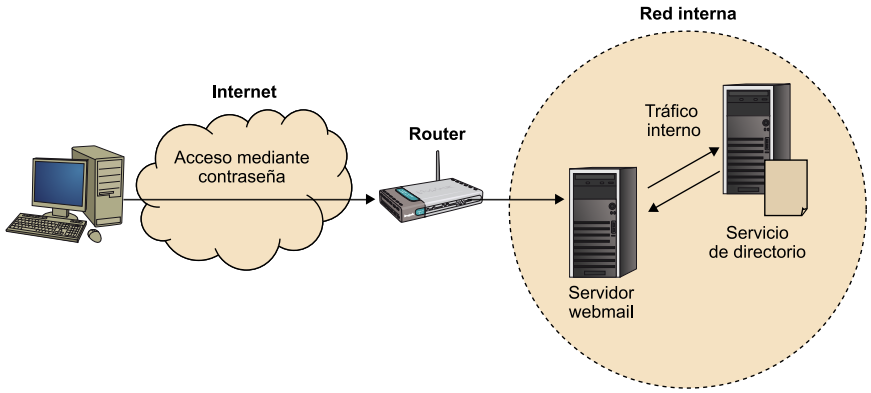
La mayoría de implementaciones de servicio de directorio permiten la autenticación del cliente (recordemos que puede ser un usuario o bien otra aplicación telemática) mediante un certificado electrónico. Otra forma de restringir los datos que contiene un servicio de directorio es haciendo que estos estén aislados de los usuarios/equipos que no pueden acceder al servicio.

Ejemplo

Si un servicio de directorio está pensado para realizar autenticaciones en servidores de correo o intranets, el servidor que controla el directorio puede estar en la red interna de la organización. El portal web de la intranet, que debe estar claramente accesible desde el exterior para poder ofrecer sus servicios, hará una consulta a una máquina (el servidor del directorio) que puede perfectamente estar ubicada en la red interna. Otra alternativa será proteger el servidor de directorio mediante un cortafuegos que permita el acceso a determinados perfiles de dirección de red.

En la figura siguiente, el cliente se conecta a un servidor webmail a través de Internet, y para autenticarse manda la contraseña. La comprobación de la contraseña se hace a nivel interno mediante el servicio de directorios que se halla sin conexión a la red exterior.

Ubicación del servicio de directorio en una red interna

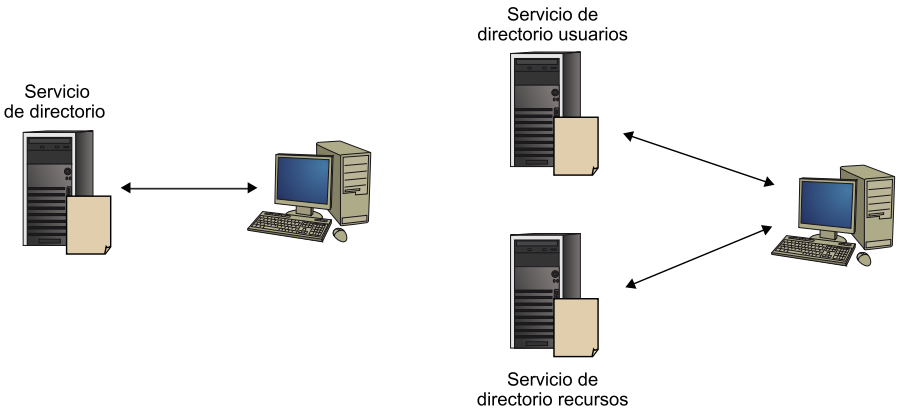


2.3.2. Eficiencia y disponibilidad

Por otra parte, tan importante como la seguridad es la eficiencia y la disponibilidad. Cierto es que un pequeño directorio para una organización pequeña podrá ejecutarse en una única máquina, de la cual se podría hacer una copia de seguridad diaria de los datos. Ahora bien, si el número potencial de peticiones de información al servicio de directorio es muy elevado, puede ser conveniente que, por motivos de eficiencia, los datos se encuentren distribuidos en más de una máquina.

La figura siguiente muestra dos disposiciones de los datos de un servicio de directorio. A la izquierda, un único servicio de directorio alberga toda la información de la organización. A la derecha son dos los servicios de directorio existentes: uno especializado en albergar la información sobre usuarios y el otro especializado en información sobre recursos.

Ejemplo de distribuciones única (izquierda) y distribuida (derecha)



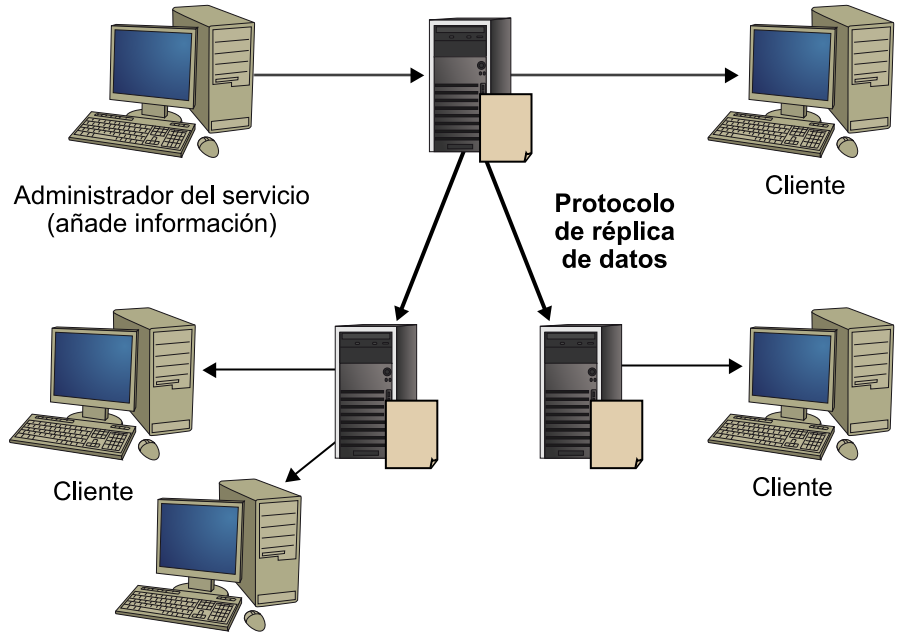
La compartición de información debería hacerse de modo que el servidor correspondiente a usuarios y el servidor correspondiente a recursos reciban aproximadamente la misma carga de operaciones. Si no puede ser así, se debería apostar por la replicación de la información en más de una máquina.

La replicación de información, es decir, el proceso de mantener múltiples copias de datos del directorio en distintas ubicaciones, propicia una serie de ventajas relacionadas con la eficiencia del sistema. En primer lugar, no se colapsará un único servicio con las peticiones de los clientes. La replicación de datos permite la existencia de un servidor que centralice las peticiones de los clientes y, en función de distintos parámetros, redirigir la petición. Por ejemplo, si se usa como parámetro la carga de trabajo de los servidores de réplica, se redirigirá la petición a la máquina que tenga menos trabajo en aquel momento. Además, si la red es relativamente grande (por ejemplo, red de campus o metropolitana) el acceso a datos será más rápido si estos se encuentran replicados cerca de la ubicación de red del cliente.

Adicionalmente, se resuelven temas de disponibilidad: si uno de los servidores que alberga la réplica cae, el cliente podrá hacer uso de alguno de los servidores de réplica que estén activos.

Para que el sistema de replicación funcione, tal como se muestra en la figura siguiente, es preciso que se contemple un protocolo de replicación de datos para dar consistencia a la información contenida en los distintos servidores.

Ejemplo de replicación de datos



3. Implementaciones de servicio de directorio

Después de realizar una aproximación teórica al concepto, diseño y usos de los servicios de directorio, y de haber estudiado el sistema LDAP, vamos a estudiar tres casos concretos de implementación:

- el OpenLDAP,
- el Apache Directory Server y
- el Active Directory.

Todos ellos están inspirados por LDAP.

3.1. OpenLDAP

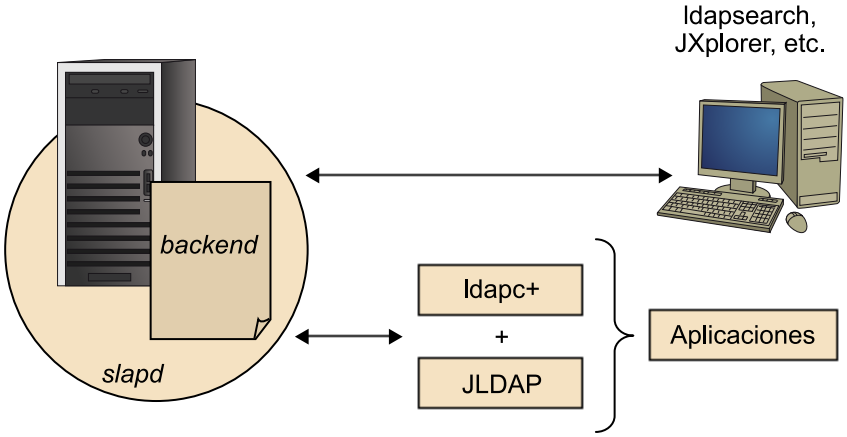
OpenLDAP es una implementación de un servicio de directorio basado en LDAP bajo la filosofía de software libre y código abierto. El proyecto OpenLDAP es quien se encarga de su desarrollo y sus productos se hallan disponibles en multitud de distribuciones GNU/Linux. Además, existen versiones para otras plataformas de sistemas operativos, como el Mac OS X o MS Windows.

Origen de OpenLDAP

OpenLDAP se desarrolló a finales de los años noventa a partir de la implementación de LDAP que hizo la Universidad de Michigan, que, como se apuntó al principio de este módulo, es la responsable de la creación de este estándar de servicios de directorio.

OpenLDAP y sus proyectos derivados proporcionan distintos componentes, algunos de ellos relacionados en el siguiente esquema.

Componentes de aplicaciones y herramientas relacionadas con LDAP



El servidor **slapd** ejecuta las funciones de servicio de directorio. Se encarga de interaccionar con el *backend* correspondiente. JLDAP y ldapc++ proporcionan bibliotecas para que puedan programarse aplicaciones en Java y en C++ respectivamente.

Para el *backend*, es decir, el almacén de datos, hay tres tipos de categorías:

1) *Backend* de almacenamiento de datos. Estos sistemas realmente almacenan información. El primero de ellos fue el back-bdb. También existe el back-ldif, que, en lugar de guardar los datos en un formato propio, los almacena directamente en ficheros de texto plano en formato LDIF.

Lectura complementaria

Las especificaciones técnicas del LDAP, *data interchange format* (LDIF) están recogidas en el RFC2849.

2) *Backend* intermediario. Actúan de *Proxy* entre el servicio de directorio LDAP y el *backend* que realmente almacena los datos. Por lo tanto se podría tener un *backend* local que realmente curse peticiones sobre un *backend* externo. Por ejemplo, el back-sql establece conexiones hacia bases de datos conformes con SQL.

SQL

Structured query language (SQL) es un sistema estándar de instrucciones para definir, gestionar y consultar datos en una base de datos relacional.

3) *Backend* dinámicos. El propósito de este tipo de *backends* es variado. Por ejemplo, los hay que están especializados en localización de servicios LDAP mediante DNS (*back-dnssrv*), los hay que permiten realizar estadísticas del uso de slapd (*back-monitor*), etc.

Además, como complemento a los distintos *backend* disponibles, existe la figura del *overlay*. Un *overlay* es un elemento que permite incrementar las funcionalidades del sistema, a través de añadir módulos.

Ejemplo

Si se precisa registrar la actividad de peticiones se puede usar el *overlay auditlog* para que se generen ficheros de texto plano o bien *accesslog* para que la actividad quede registrada usando una base de datos LDAP.

OpenLDAP permite la replicación de contenidos. En las primeras versiones se utilizaban los conceptos de servidores máster y esclavos: el máster aceptaba actualizaciones por parte de los clientes, mientras que los esclavos sólo aceptaban actualizaciones desde un máster. Recordemos que la actualización o sincronización de contenidos entre los distintos servidores es esencial para el buen funcionamiento de la replicación de contenidos. Actualmente, se utilizan los términos de *proveedor* y *consumidor de actualizaciones*. Esto permite definir mejores reglas de actualización, haciendo posible que un servidor pueda actuar de proveedor o bien de consumidor en función de la necesidad.

Lectura complementaria

En la RFC 4533 se detallan los protocolos de sincronización.

El motor de sincronización para OpenLDAP es **syncrepl**, que usa como protocolo el **LDAP Sync**. Este protocolo permite tanto actualizaciones tipo *pull*, donde el consumidor hace encuestas periódicas al proveedor para ver si ha habido actualizaciones, o tipo *push*, donde el consumidor está atento a las notificaciones de actualización mandadas por parte del proveedor.

3.2. *Apache directory server*

OpenLDAP es uno de los servicios de directorio más utilizados en la actualidad, aunque también hay otras alternativas atractivas de software libre, como por ejemplo el servicio de directorios que contempla Apache.

El *Apache directory server* (ApacheDS) es un servicio de directorio desarrollado con el lenguaje de programación Java y disponible bajo la licencia Apache Software. También puede funcionar como servidor de autenticación Kerberos, pero su uso principal es de servidor LDAP.

Ved también

El servidor de autenticación Kerberos se estudia en el módulo "*Single sign-on* y federación de identidades".

El sistema puede empotrarse sin dificultad en otros proyectos basados en componentes Java, lo cual lo hace atractivo como motor de servicio de directorio si se está desarrollando un sistema de información basado en Java. En este sentido, si OpenLDAP está más indicado para funcionar como gran servicio de directorio, que pueda estar replicado y distribuido, y funcione como un servicio individual, ApacheDS está más pensado para formar parte de un paquete de aplicaciones que precisen, en principio internamente, de un servicio de directorio.

Las aplicaciones Java usan JNDI[7] para interactuar con el servicio de directorio. De hecho, ApacheDS implementa un JNDI.

(7)JNDI son las siglas de *Javanaming and directory interface.*

ApacheDS utiliza una serie de interfaces llamadas *multipurpose interfaces for network applications* (MINA). Estas interfaces contienen métodos para generar objetos que implementen nuevas funcionalidades. Así pues, se pueden implementar protocolos que usen ApacheDS como medio de almacenaje y gestión.

Finalmente, al igual que OpenLDAP, Apache DS tiene una serie de "interceptores" y otros elementos de software que permiten su extensión.

3.3. Active Directory

En los años noventa, Microsoft implantó el concepto de sistema operativo de red en sus productos. En concreto, Windows NT 3.0 ya implementaba un entorno en red, con distintos recursos accesibles remotamente y usuarios, gestionados por un administrador. El concepto de dominio agrupaba distintos recursos y usuarios.

A finales de los noventa, apareció el Active Directory, una concreción del concepto de directorio adaptado mediante técnicas provenientes del LDAP para poder dar respuesta a propiedades de robustez y rendimiento.

Si Windows NT usaba el NetBIOS como mecanismo primario de comunicación de red (y el WINS como base de datos de nombre de objetos de red), el Active Directory requiere el uso de TCP/IP, así como del servicio DNS.

Al igual que otras implementaciones de LDAP, ofrece un sistema de administración centralizado, y la organización de los datos en forma distribuida y replicada.

Active Directory incorpora un almacén de datos para guardar información sobre los objetos (por ejemplo, objetos como usuarios, impresoras, etc.). La estructura que ofrece se basa en cuatro conceptos:

1) El dominio es la estructura básica, la cual agrupa todos los objetos que se administran. Un dominio se puede identificar mediante una estructura DNS.

2) La unidad organizativa es una unidad inferior, que puede estar compuesta por otras unidades organizativas, pero también por grupos y objetos.

3) Los grupos son conjuntos de objetos del mismo tipo.

4) Finalmente, la unidad básica es el objeto, es decir, la representación de los recursos y usuarios del sistema en red.

En caso de que haya varios dominios compartiendo un espacio de nomenclaturas y un esquema común, se establece la estructura de **árbol de dominios**.

Ejemplo de árbol de dominios

Podría haber un árbol formado por los dominios siguientes:

- hospitalCiudad1.hospitales.org
- hospitalCiudad2.hospitales.org
- hospitalCiudad3.hospitales.org

Finalmente, un **bosque** es una colección de árboles que, aunque no comparten un espacio de nomenclatura contiguo, tienen un esquema común. Por ejemplo: hospitales.org, ayuntamientos.org, etc.

En un sistema basado en Active Directory, un servidor puede desempeñar varios papeles:

- **Controlador de dominio**. Estos servidores pertenecen al dominio y contienen una copia de los datos pertenecientes a recursos y usuarios. Contienen una copia de la cuenta del usuario. Son un elemento indispensable del dominio y se pueden usar varios de ellos para distribuir y/o replicar la información.

- **Servidor de catálogo global**. Incluyen información sobre todos los objetos de un bosque.

- **Servidor miembro**. Pertenecen también al dominio, pero no contienen copias de las cuentas de usuario. Se usan para guardar los archivos y recursos de red.

- **Servidor independiente**. Este servidor no tiene nada que ver con la gestión de Active Directory (por ejemplo, un servidor Windows que pertenezca a un grupo de trabajo concreto).

Cuando hay varios servidores cohabitando para la replicación o distribución de información, es conveniente usar un esquema de red para poder optimizar la gestión del tráfico entre distintos servidores. Ahora bien, operaciones como la modificación del esquema sólo pueden realizarse por uno de los servidores, y no desde cualquiera del sistema de réplicas.

Active Directory en sistemas Unix

Pese a ser un producto de Microsoft para plataformas Windows, Active Directory puede integrarse en sistemas de información basados en sistemas operativos tipo Unix mediante software de terceras partes.

Resumen

En este módulo hemos estudiado el concepto de servicio de directorio y hemos visto aspectos relacionados con su diseño e implantación.

En primer lugar hemos estudiado el uso de los directorios. Se ha visto que son un tipo concreto de base de datos que nos devuelve información a partir de la identificación de una entrada (el DN) o bien a partir de búsquedas en el directorio. Hemos revisado distintos tipos de directorios, desde los integrados en una aplicación hasta los directorios de sistemas operativos de red. Hemos visto la relación de un servicio de directorio con la seguridad de la información y el acceso a aplicaciones. Hemos descrito varios ejemplos de espacio de nombres, en concreto el DNS, el sistema WINS y el espacio de nombres descrito para LDAP. Hemos estudiado el concepto de *distinguished name*. Hemos concluido esta parte del módulo describiendo cuáles son las operaciones que los clientes pueden realizar con un servicio de directorio. Hemos visto de forma detallada las operaciones de interrogación, en especial la búsqueda. Asimismo, se han indicado las herramientas que existen para que otras aplicaciones y programas puedan acceder a los datos que contiene un directorio.

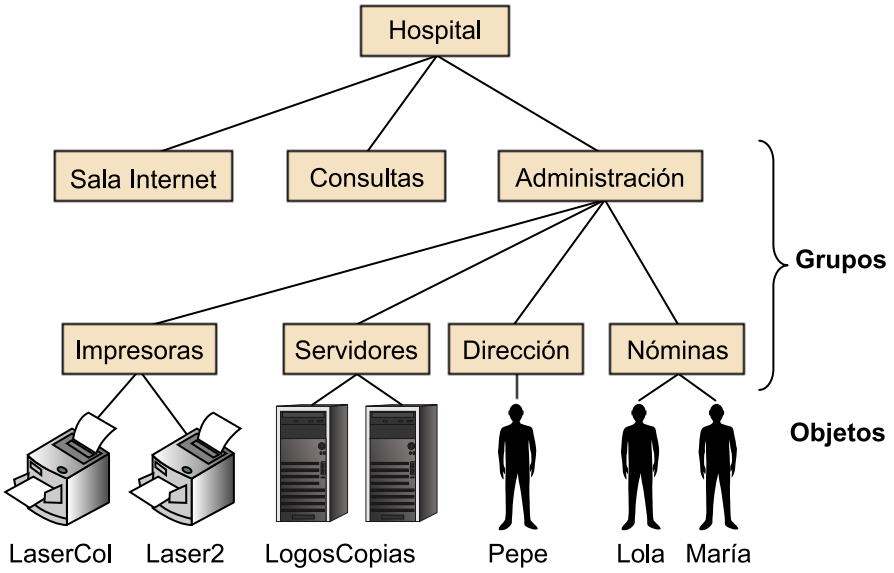
El siguiente apartado lo hemos dedicado a aspectos de diseño del directorio. En referencia al diseño del espacio de nombres, hemos tratado la elección del sufijo del directorio (cómo empieza el espacio de nombres), cómo decidir la estructura (hemos visto que hay espacios de nombres planos y jerárquicos), y cómo identificar los objetos. También hemos profundizado en el concepto de esquema del directorio, mediante el cual describimos qué información se guarda en el directorio. Hemos expuesto cómo se define un atributo en LDAP y también cómo podemos definir una clase de objeto. Para finalizar este apartado, hemos discutido aspectos relacionados con la seguridad en el acceso al directorio, la eficiencia del servicio de directorio y la disponibilidad de los datos que éste contiene.

Para finalizar el módulo, hemos descrito tres implementaciones de servicio de directorio basadas en LDAP y ampliamente utilizadas: el OpenLDAP, el Apache Directory Server y el Active Directory de Microsoft.

Actividades

1. En la figura siguiente se muestra la estructura de un directorio de un hospital, que contempla la red con los recursos y usuarios. Está dividido en tres unidades organizativas: una sala de Internet, los recursos informáticos de las consultas del hospital y los recursos informáticos de la administración. En concreto, la unidad organizativa de "Administración", incluye el grupo "Dirección" (integrado sólo por un usuario), el grupo "Nóminas", integrado por los usuarios que pertenecen a la gestión de nóminas, un grupo "Servidores" con un par de máquinas y un par de impresoras, ubicadas dentro del grupo "Impresora".

Escenario de directorio



Diseñad un esquema para esta organización, detallando qué clases y atributos vais a utilizar.

2. Utilizando el esquema anterior, cread algunas entradas al directorio. Es decir, para los usuarios y recursos de la figura, escribid los atributos y sus valores, que se almacenarían en el directorio.

3. Suponed que el usuario Pepe también debe pertenecer al grupo de nóminas. Detallad la solución, es decir, los atributos y sus valores, que se deberían almacenar en el directorio para los usuarios de la figura anterior.

4. Obtened e instalad un servidor de directorios basado en LDAP (OpenLDAP sería un buen candidato). Mediante la herramienta Jxplorer, acceded a este servidor e intentad implementar la solución propuesta en las actividades anteriores.

5. Usando un lenguaje de programación que conozcais, elaborad una pequeña aplicación que permita listar entradas del directorio en función de parámetros variados, como por ejemplo, la clase de objeto de las entradas a listar.

Glosario

alias *m* Entrada que en realidad enlaza con otra entrada del directorio.

atributo *m* Parte de la entrada destinada a guardar una pieza de información, como por ejemplo un apellido o un número de teléfono.

base *f* En la operación de búsqueda especifica, desde qué entrada u objeto se quiere empezar a buscar información.

búsqueda *f* Operación básica de interrogación al servicio del directorio, mediante la cual se obtiene información conforme una serie de criterios.

DAP (*directory access protocol*) *m* Sistema primitivo de implementación y gestión de directorios, basado en la especificación X.500.

directorio *m* Tipo especializado de base de datos jerárquica que organiza y almacena datos acerca de elementos (entradas de directorio).

***distinguished name* (DN)** *m* Relación de DN relativos que identifican unívocamente una entrada en un directorio LDAP.

DN relativo *m* Atributo y su valor que identifican a una entrada para un nivel en concreto del árbol del directorio.

esquema *m* Definición de qué tipos de objeto guarda un directorio y los atributos usados en la definición de los objetos.

LDAP (*lightweight DAP*) *f* Interfaz entre clientes de directorio y sistemas DAP, que luego evolucionó hacia un servicio de directorio.

objectclass *m* Atributo de una entrada de directorio basado en LDAP que especifica a qué clase pertenece la entrada (por ejemplo, "person").

servicio de directorio *m* Plataforma que proporciona métodos para gestionar y almacenar los datos que contiene el directorio.

sufijo *m* DN para la raíz del árbol de directorio.

user identifier *m* Atributo específico para guardar el "login" de un usuario de un sistema informático.

Bibliografía

Desmond, R. y otros (2008). *Active Directory: Designing, Deploying and Running Active Directory*. Sebastopol (California): O'Reilly.

Howes, T. A. y otros (2003). *Understanding and Deploying LDAP Directory Services* (2.ª ed.). Boston (Massachusetts): Addison-Wesley Longman.

Raya, J. L. y otros (2008). *Aprenda Microsoft Windows Server 2008*. Madrid: Ra-Ma.