OpenLDAP Wrapper Tutorial

Diego S. Marafetti **<*dmarafetti@gmail.com*>**

Historial de Revisiones

Revisión 1	16/04/2009	Borrador inicial

Contenidos del documento

Introducción

El objetivo de este documento es explicar el uso de la API para el manejo del wrapper de openLDAP. Esta API pretende introducir una capa de abtracción sobre la API de OpenLDAP de manera de reducir la complejidad en el uso de la misma. Está dirigido a los estudiantes que cursan Sistemas Operativos y que no poseen los conocimientos de LDAP.
Es muy importante tener en claro que esta API de ninguna manera pretende ser el reemplazo que el usuario deberá utilizar por la biblioteca original de OpenLDAP. Lo que si pretende es:

- ✓ Darle al usuario un codigó más simple que le permite realizar las operaciones más básicas sobre un servidor LDAP.
- ✓ Mostrar que funciones de openLDAP tiene que utilizar para cada operación.
- ✓ Para el usuario que tiene poco conocimiento de programación en C y quiere reforzar los conceptos de manejo de estructuras de datos y organización de código.

Tipo de datos

`LDAP_CONTEXT`	Esta estrucutra representa un contexto de ejecución sobre un servidor LDAP. Contiene el host donde se encuentra el servidor y una referencia al motor interno de openLDAP. A partir del contexto se pueden crear sesiones.
`LDAP_SESSION`	Una sesión estará compuesta por un usuario ejecutando operaciones sobre el directorio. Para iniciar una sesion es necesario registrar el nombre del directorio a operar (DN) y el password.
`LDAP_ENTRY`	Representa una entrada del directorio DN. Este objeto representa una entidad del mundo real. Cada una posee el nombre que la identifica (dn) y una lista de atributos. Se utliza para realizar las operaciones de add, delete, modify.
`PLDAP_ATTRIBUTE`	Un atributo pertenece a una entry. Este posee un nombre y un conjunto de valores.
`LDAP_RESULT_SET`	En una operación de búsqueda este objeto representa el conjunto de resultados (records) que se obtuvieron de una consulta a un directorio.
`LDAP_RECORD`	Un record representa a una entry en un record set.
`LDAP_FIELD`	Representa a un atributo de una entry y se tiene acceso al nombre y a sus valores.
`LDAP_ITERATOR`	Esta estructura mantiene el estado de un record set para ser utilizada por las operaciones del iterator para poder recorrer un record set.

El wrapper también define ciertas estructuras que representan operaciones sobre las entidades descritas más arriba. En otras palabras son structs con punteros a funciones. Esto permite el acceso a esas funciones desde un punto conocido y agruparlas según sus funciones.

`LDAP_ATTRIBUTE_OP`	Operaciones sobre atributos.
`LDAP_ENTRY_OP`	Operaciones sobre entry.

LDAP_RECORD_OP	Operaciones sobre records.
LDAP_SESSION_OP	Operaciones sobre una session.
LDAP_CONTEXT_OP	Operaciones sobre un contexto.

Modo de uso

Conociendo los parametros de conexión se crea un nuevo contexto de ejecución. Luego se crea una nueva session para comenzar con las operaciones.

```
PLDAP_SESSION       session;
PLDAP_CONTEXT       context      = newLDAPContext();
PLDAP_CONTEXT_OP    ctxOp        = newLDAPContextOperations();
PLDAP_SESSION_OP    sessionOp    = newLDAPSessionOperations();

ctxOp->initialize(context, "ldap://192.168.1.3:1389");

session = ctxOp->newSession(context, "cn=Directory Manager", "mipassword");


se inicia la session. Se establece la
conexión con el servidor LDAP

sessionOp->startSession(session);



 ===   las operaciones se realizan aqui... ===



finaliza la session con el servidor

sessionOp->endSession(session);


funciones de utilería para liberación de
la memoria alocada por las estructuras

freeLDAPSession(session);
freeLDAPContext(context);
freeLDAPContextOperations(ctxOp);
freeLDAPSessionOperations(sessionOp);
```

Agregar una nueva entry

El ejemplo muestra que se crea una nueva entry usando las funciones definidas en las operaciones pertenecientes a las entry. Esto retorna la referencia a una nueva entry alocada. Queda en responsabilidad del programador en liberar la memoria. Luego se le asigna el identificador único a la entry.
A continuación se crea de forma similar un atributo de la entry con sus valores. Esa función acepta un número variable de argumentos.

```
PLDAP_ENTRY entry = entryOp->createEntry();

entry->dn = "utnurlID=00009,ou=so,dc=utn,dc=edu";

entryOp->addAttribute(entry, attribOp->createAttribute("objectclass", 2, "top", "utnUrl"));

sessionOp->addEntry(session, entry);
```

Modificar una entry

La forma es similar al ejemplo anterior pero seleccionando la edición del nuevo atributo (ó de uno existente) con el atributo y sus respectivos nuevos valores.

```
entryOp->editAttribute(entry, attribOp->createAttribute("utnurlTitle", 1, "Pagina de Prueba
para LDAP"));

sessionOp->editEntry(session, entry);
```

Eliminar una entry

Se ofrecen dos maneras de eliminar una entry. La primera es simplemente conociendo el dn y la otra es teniendo el objeto entry. Es muy sencillo.

```
sessionOp->deleteEntryObj(session, entry);

sessionOp->deleteEntryDn(session, "utnurlID=00009,ou=so,dc=utn,dc=edu");
```

Búsquedas

Una búsqueda se inicia llamando a la siguiente función que retorna un record set como resultados. Para ello es necesario indicar la rama del árbol en la cual se realizará la búsquda y la expresión de filtro. En este caso se búscar todas las entry que posean cualquier valor en el atributo utnurlKeywords.

```
PLDAP_RESULT_SET resultSet = sessionOp->searchEntry(session, "ou=so,dc=utn,dc=edu",
"utnurlKeywords=*");
```

Para poder recorrer los resultados obtenidos se puede instanciar un iterador. Este iterador retorna la siguiente entrada y solo puede ser recorrida una vez en la versión de la API actual. Con esto se tiene acceso a la referencia del record (una entry) sobre la iteración actual.

```
resultSet->iterator

iterator->hasNext(resultSet);

PLDAP_RECORD record = iterator->next(resultSet);
```

Siguiendo el mismo patrón de iteración un record también expone funciones para iterar sobre sus atributos. Accediendo a un field se tiene acceso al nombre y a sus valores actuales. Luego es necesario liberar la memoria de las estructuras creadas.

```
PLDAP_FIELD field = recordOp->nextField(record);

field->name

field->valuesSize
```