NAME

ldap_init, ldap_initialize, ldap_open - Initialize the LDAP library and open a connection to an LDAP server

LIBRARY

OpenLDAP LDAP (libldap, -lldap)

SYNOPSIS

#include <ldap.h>

LDAP *ldap_open(host, port)
char *host;
int port;

LDAP *ldap_init(host, port)
char *host;
int port;

int ldap_initialize(ldp, uri)
LDAP **ldp;
char *uri;

int ldap_connect(ldp)
LDAP *ldp;

int ldap_set_urllist_proc(ld, proc, params)
LDAP *ld;
LDAP_URLLIST_PROC *proc;
void *params;

int (LDAP_URLLIST_PROC)(ld, urllist, url, params);
LDAP *ld;
LDAPURLDesc **urllist;
LDAPURLDesc **url;
void *params;

#include <openldap.h>

int ldap_init_fd(fd, proto, uri, ldp)
ber_socket_t fd;

```
int proto;
char *uri;
LDAP **ldp;
```

DESCRIPTION

ldap_open() opens a connection to an LDAP server and allocates an LDAP structure which is used to identify the connection and to maintain per-connection information. **ldap_init()** allocates an LDAP structure but does not open an initial connection. **ldap_initialize()** allocates an LDAP structure but does not open an initial connection. **ldap_init_fd()** allocates an LDAP structure using an existing connection on the provided socket. One of these routines must be called before any operations are attempted.

ldap_open() takes *host*, the hostname on which the LDAP server is running, and *port*, the port number to which to connect. If the default IANA-assigned port of 389 is desired, LDAP_PORT should be specified for *port*. The *host* parameter may contain a blank-separated list of hosts to try to connect to, and each host may optionally by of the form *host:port*. If present, the *:port* overrides the *port* parameter to **ldap_open()**. Upon successfully making a connection to an LDAP server, **ldap_open()** returns a pointer to an opaque LDAP structure, which should be passed to subsequent calls to **ldap_bind()**, **ldap_search()**, etc. Certain fields in the LDAP structure can be set to indicate size limit, time limit, and how aliases are handled during operations; read and write access to those fields must occur by calling **ldap_get_option**(3) and **ldap_set_option**(3) respectively, whenever possible.

ldap_init() acts just like **ldap_open()**, but does not open a connection to the LDAP server. The actual connection open will occur when the first operation is attempted.

ldap_initialize() acts like **ldap_init()**, but it returns an integer indicating either success or the failure reason, and it allows to specify details for the connection in the schema portion of the URI. The *uri* parameter may be a comma- or whitespace-separated list of URIs containing only the *schema*, the *host*, and the *port* fields. Apart from **ldap**, other (non-standard) recognized values of the *schema* field are **ldaps** (LDAP over TLS), **ldapi** (LDAP over IPC), and **cldap** (connectionless LDAP). If other fields are present, the behavior is undefined.

At this time, **ldap_open()** and **ldap_init()** are deprecated in favor of **ldap_initialize()**, essentially because the latter allows to specify a schema in the URI and it explicitly returns an error code.

ldap_connect() causes a handle created by **ldap_initialize()** to connect to the server. This is useful in situations where a file descriptor is required before a request is performed.

ldap_init_fd() allows an LDAP structure to be initialized using an already-opened connection. The *proto* parameter should be one of LDAP_PROTO_TCP, LDAP_PROTO_UDP, or

LDAP_PROTO_IPC for a connection using TCP, UDP, or IPC, respectively. The value LDAP_PROTO_EXT may also be specified if user-supplied sockbuf handlers are going to be used. Note that support for UDP is not implemented unless libldap was built with LDAP_CONNECTIONLESS defined. The *uri* parameter may optionally be provided for informational purposes.

ldap_set_urllist_proc() allows to set a function *proc* of type *LDAP_URLLIST_PROC* that is called when a successful connection can be established. This function receives the list of URIs parsed from the *uri* string originally passed to **ldap_initialize()**, and the one that successfully connected. The function may manipulate the URI list; the typical use consists in moving the successful URI to the head of the list, so that subsequent attempts to connect to one of the URIs using the same LDAP handle will try it first. If *ld* is null, *proc* is set as a global parameter that is inherited by all handlers within the process that are created after the call to **ldap_set_urllist_proc()**. By default, no *LDAP_URLLIST_PROC* is set. In a multithreaded environment, **ldap_set_urllist_proc()** must be called before any concurrent operation using the LDAP handle is started.

Note: the first call into the LDAP library also initializes the global options for the library. As such the first call should be single-threaded or otherwise protected to insure that only one call is active. It is recommended that **ldap_get_option**() or **ldap_set_option**() be used in the program's main thread before any additional threads are created. See **ldap_get_option**(3).

ERRORS

If an error occurs, **ldap_open()** and **ldap_init()** will return NULL and *errno* should be set appropriately. **ldap_initialize()** and **ldap_init_fd()** will directly return the LDAP code associated to the error (or *LDAP_SUCCESS* in case of success); *errno* should be set as well whenever appropriate. **ldap_set_urllist_proc()** returns LDAP_OPT_ERROR on error, and LDAP_OPT_SUCCESS on success.

SEE ALSO

ldap(3), **ldap_bind**(3), **ldap_get_option**(3), **ldap_set_option**(3), **lber-sockbuf**(3), **errno**(3)

ACKNOWLEDGEMENTS

OpenLDAP Software is developed and maintained by The OpenLDAP Project <http://www.openldap.org/>. **OpenLDAP Software** is derived from the University of Michigan LDAP 3.3 Release.