参数：

ld：连接句柄。

dn：要修改的条目名。

mods：设置修改条目的空结尾的数组。此参数在 LDAPMod 结构中有如下意义。

mod_op：执行的修改操作，为 LDAP_MOD_ADD、LDAP_MOD_DELETE、LDAP_MOD_REPLACE 之一。此字段也指明在 mod_vals 联合中的值的类型。此字段同 LDAP_MOD_BVALUES 进行或操作，以指定 mod_bvalues 值的形式。另外，也可以使用 mod_values 形式。

mod_type：要修改的类型。

mod_vals：对此值进行 add、delete、replace 操作。仅 mod_values 或 mod_bvalues 之一可以使用。mod_values 为以 NULL 结尾的字符串数组，mod_bvalues 为以 NULL 结尾的 berval 结构数组，可以用来传送图像一类的二进制值。

对于 LDAP_MOD_ADD 操作，给定的值是要添加的条目，根据需要创建属性。

对于 LDAP_MOD_DELETE 操作，给定的值要从条目中删除，如果条目属性被删除，mod_vals 字段应设为 NULL。

对于 LDA_REPLACE 操作，对属性进行替换，或根据需要创建。所有的修改操作按列出的顺序执行。

ldap_modify_s 根据修改操作结果返回 LDAP 错误码。此代码由 ldap_perror 及相关函数解释。ldap_modify 返回请求初始化消息 ID，或出错时返回−1。操作结果由 ldap_result 获得。

（8）修改条目的 RDN

ldap_modrdn 和 ldap_modrn_s 函数用来修改 LDAP 条目名。

```
int ldap_modrdn( LDAP   *ld, char   *dn, char   *newrdn, int deleteoldrdn);
int ldap_modrdn_s(LDAP *ld, char   *dn, char   *newrdn, int deleteoldrdn );
```

参数：

ld：连接句柄。

dn：要修改的条目名。

newrdn：条目的新的 RDN。

deleteoldrdn：布尔值，非 0 值指明删除旧的 RDN，0 值指明旧的 RDN 应保留为条目的非区别值。

ldap_modrdn_s 函数为同步调用，返回指明操作输出的 LDAP 错误码。ldap_modrdn 函数为异步调用，返回初始化操作的消息 ID，或者出错返回−1。由 ldap_result 取得返回结果。

（9）添加条目

ldap_add 和 ldap_add_s 用来添加 LDAP 目录条目。

```
int ldap_add( LDAP *ld, char *dn, LDAPMod *attrs[] );
int ldap_add_s( LDAP *ld, char *dn, LDAPMod *attrs[] );
```

参数：

ld：连接句柄。

dn：要添加的条目名。

attrs：条目属性，使用为 ldap_modify 定义的 LDAPMod 结构。mod_type 和 mod_vals 字段需要填充，mod_op 字段会被忽略，除非同 LDAP_MOD_BVALUES 进行逻辑或运算（|）后，用来指定值为 mod_vals 中的 mod_bvalues。

注意：条目的父条目必须存在。

ldap_add_s 为同步函数，返回指明操作输出的 LDAP 错误码。ldap_add 为异步函数，返回操作初始化消息 ID，或出错时返回−1，结果由 ldap_result 函数取得。

（10）删除条目

ldap_delete 和 ldap_delete_s 用来从 LDAP 目录中删除条目。

```
int ldap_delete(LDAP* ld, char* dn);
int ldap_delete_s(LDAP* ld, char* dn);
```

参数：

ld：连接句柄。

dn：要删除的条目名。

注意：要删除的条目必须为叶子条目（例如，它不能有子条目）。LDAP 不支持删除条目子树的操作。

ldap_delete_s 为同步函数，返回指明操作输出的 LDAP 错误代码。ldap_delete 为异步函数，返回操作初始化消息 ID，或出错时返回−1，结果由 ldap_result 函数取得。

2．操作放弃

ldap_abandon 用来放弃一个操作过程。

```
int ldap_abandon(LDAP* ld, int msgid);
```

ldap_abandon 放弃消息 ID 为 msgid 的操作。如果操作成功返回 0，否则返回−1。成功调用 ldap_abandon 后，给定消息 ID 的结果不会由 ldap_result 调用返回。

3．取得结果的调用

ldap_result 用来取得先前同步初始化的结果。ldap_msgfree 用来释放先前调用 ldap_result 或同步查询函数取得的结果。

```
int ldap_result(LDAP *ld, int   msgid, int   all, struct timeval   *timeout, LDAPMessage   **res);
int ldap_msgfree( LDAPMessage *res );
```

参数：

ld：连接句柄。

msgid：需要返回结果的操作的消息 ID。

all：布尔值，代表查询结果的含义，非 0 值指明在所有查询结果都应取得后才能返回。如为 0，查询结果（条目）将会一次返回查到的一个。

timeout：表示等待返回结果的超时时间。NULL 值将造成 ldap_result 阻塞等待，直到结果可用。timeout 值为 0 秒表示轮询状态。

res：对于 ldap_result，是一个包含操作结果集的结果参数。对于 ldap_msgfree，表示要被释放的结果参数从先前的 ldap_result、ldap_search_s 或 ldap_ seatch_st 调用取得。

在成功完成后，ldap_result 返回结果的类型，这些类型为以下常量：LDAP_RES_BIND，LDAP_RES_SEARCH_ENTRY，LDAP_RES_SEARCH_RESULT，LDAP_RES_MODIFY，LDAP_RES_ADD，LDAP_RES_DELETE，LDAP_RES_MODRDN，LDAP_RES_COMPARE。

ldap_result 在超时后返回 0，出错后返回−1。在这种情况下，ld 结构的 ld_err 字段会相应设置。

ldap_msgfree 释放指向 res 的结果结构并返回释放的消息类型。

4．错误处理调用

下面的调用捕获其他 LDAP API 返回的错误。

```
int ldap_result2error( LDAP *ld, LDAPMessage    *res, int    freeit );
char* ldap_err2string(int err);
void ldap_perror(LDAP* ld, char* msg);
```

参数：

ld：连接句柄。

res：ldap_result 返回的 LDAP 操作结果，或一个异步 API 操作调用的结果。

freeit：布尔值，指明 res 参数是否被释放（非 0 值释放，0 值不释放）。

err：LDAP 错误码。

msg：在 LDAP 错误信息之前显示的信息。

ldap_result2error 用来将 res 参数转换成数字形式的错误码。同时也将结果信息中的 ld_matched 和 ld_error 部分解析出来，并将它们放入连接句柄信息中。所有的同步操作函数在返回前调用 ldap_result2error，确保这些字段正确设置。

连接结构的相关字段如下。

ld_matched：此参数包含 LDAP 服务器返回结果的错误信息。

ld_errno：指明操作 LDAP 错误码。

ldap_err2string 用来将 LDAP 错误码转换成以 NULL 结尾的包含错误描述的字符串。此错误码为 ldap_result2error 或一个同步 API 操作调用的返回结果。

5．解释查询条目的调用

以下函数用来解析 ldap_search 及相关函数返回的条目。这些条目返回一个只应被下面函数访问的非透明的结构。这些函数提供遍历所返回的条目，遍历条目的属性，取得条目名称，取得条目的给定属性的属性值。

（1）遍历条目集

ldap_first_entry 和 ldap_next_entry 函数用来遍历查询结果的条目集。

ldap_count_entries 用来计算返回的条目数量。

```
LDAPMessage* ldap_first_entry(LDAP* ld, LDAPMessage* res);
LDAPMessage* ldap_next_entry(LDAP* ld, LDAPMessage* entry);
```

```
int ldap_count_entry(LDAP* ld, LDAPMessage* res);
```

参数：

ld：连接句柄。

res：查询结果。

entry：先前的 ldap_first_entry 或 ldap_next_entry 调用返回的条目。

ldap_first_entry 和 ldap_next_entry 在结果中没有条目时返回 NULL。当在遍历条目时发生错误也返回 NULL，这种情况下，ld 连接句柄的 ld_errno 字段会被设置为错误码。

ldap_count_entries 返回条目链的条目数，它也用来计算 ldap_first_entry 或 ldap_next_entry 返回的条目链中的剩余条目数。

（2）遍历条目属性

ldap_first_attribute 和 ldap_next_attribute 调用用来遍历一个条目的属性列表。

```
char *ldap_first_attribute(LDAP *ld, LDAPMessage   *entry, void **ptr);
char *ldap_next_attribute(LDAP *ld, LDAPMessage   *entry, void *ptr);
```

参数：

ld：连接句柄。

entry：要遍历的条目，由 ldap_first_entry 或 ldap_next_entry 返回。

ptr：在 ldap_first_attribute 中，用来内部跟踪当前条目位置的指针地址，在 ldap_next_attribute 中，由先前的 ldap_first_attribute 调用返回的指针。

ldap_first_attribute 和 ldap_next_attribute 在到达属性结尾时会返回 NULL，或者如果发生错误，ld 连接句柄的 ld_errno 字段会被设置为错误码。

两个函数返回一个指向预先连接的包含当前属性名的缓冲区，这应看作静态数据。ldap_first_attribute 将会分配空间并返回指向用来跟踪当前位置的 BerElement 类型指针。此指针应传给后面调用的 ldap_next_attribute 来遍历条目属性。

返回的属性名由 ldap_get_values 及相关函数解析并取得相关联的值。

（3）获取属性值

ldap_get_values 和 ldap_get_values_len 用来从条目中取得给定属性的值。ldap_count_values 和 ldap_count_values_len 用来计算返回值的个数。ldap_value_free 和 ldap_value_free_len 用来释放属性值。

```
typedef struct berval {
    unsigned long    bv_len;
    char *bv_val;
};
char **ldap_get_values(LDAP *ld, LDAPMessage *entry, char *attr );
struct berval **ldap_get_values_len(LDAP *ld, LDAPMessage *entry, char *attr);
int ldap_count_values( char **vals );
int ldap_count_values_len( struct berval **vals );
int ldap_value_free( char **vals );
int ldap_value_free_len( struct berval **vals );
```

参数：

ld：连接句柄。

entry：获取属性值的条目。

attr：需要获取的属性值。

vals：调用 ldap_get_values 或 ldap_get_values_len 返回的值。

上面提供了两种形式的变量调用，第一种形式仅适于非二进制的字符串数据，第二种以_len 结尾的形式可以使用任何类型的数据。

注意，返回内存在不使用时，为了防止内存泄露，不要忘记调用 ldap_value_free 或 ldap_value_free_len 释放。

（4）取得条目名称

ldap_get_dn 用来返回条目名称。

ldap_explode_dn 用来拆分名称为部分名。

ldap_dn2ufn 用来转换名称为更多的“用户友好”的格式。

```
char *ldap_get_dn( LDAP *ld, LDAPMessage *entry );
char **ldap_explode_dn( char *dn, int notypes );
char *ldap_dn2ufn( char *dn );
```

参数：

ld：连接句柄。

entry：需要取得名称的条目。

dn：要拆分的 dn，由 ldap_get_dn 返回。

notypes：布尔型参数，如果非 0，则 dn 的各部分应该取消类型名称。（例如，“cn=Babs”应变成“Babs”。）

ldap_get_dn 在解析 dn 发生错误时返回 NULL，设置 ld 连接句柄的 ld_errno 字段以指明错误。返回的指针由函数内部申请空间，所以在不用时应调用 ldap_memfree 释放。

ldap_explode_dn 返回包含 DN 的 RDN 部分字符数组，带有或不带类型由 notypes 参数指定。返回的数组在不再使用时应调用 ldap_value_free 释放。

ldap_dn2ufn 转换 DN 为用户友好的格式（参照 RFC 1781）。UFN 返回的是已分配的空间，应在不用时调用 ldap_memfree 释放。

6．简单 LDAP API 代码

```
#include <ldap.h>
int main(int argc, char *argv[])
{
    LDAP *ld;
    LDAPMessage     *res, *e;
    int   i;
    char *a, *dn;
    void *ptr;
    char **vals;
```