表 6-8　搜索选项列表

搜索选项	说　明
-a deref	指定别名反向引用。可选值为 never、always、search 或 find 如果不使用此参数，默认为 never 当服务端遇到别名时，它可选择反向引用或直接返回别名，“find”在定位基对象时进行反向引用，但对基对象的子条目则直接返回；“search”在定位基对象时不执行反向引用，但对基对象的子条目进行反向引用；“never”从不进行反向引用，直接返回条目本身；“always”总是执行反向引用
-A	只检索属性的名称，而不检索属性的值
-b basedn	指定用作搜索起始点的基对象的 DN。使用引号来指定该值，例如："ou=West, o=Acme, c=US" 如果要搜索的服务器需要指定搜索起点，则必须使用此参数。否则此参数是可选的 也可以同时使用-b 和-s 来确定搜索范围。没有–s，–b 就会搜索指定为起始点的项以及该项的所有子项
-c	遇到错误跳过，并继续执行
-f file	从文件中读取操作参数
-l limit	指定完成搜索的时间限制（秒）。如果没有指定此参数或指定的限制为 0，那么搜索就没有时间限制。但是，ldapsearch 的等待时间不会超过服务器上设置的搜索时间限制
-L	以 LDIFv1 格式输出响应
-LL	以 LDIF 格式输出响应，但不包含注释
-LLL	以 LDIF 格式输出响应，但包含注释和版本号
-P version	协议版本号，默认为 v3
-s scope	指定使用-b 参数时的搜索范围，-b 和-s 的参数出现的顺序并不重要 base—仅搜索-b 参数指定的项 one—仅搜索-b 参数指定项的直接子项，而不搜索该项本身 sub—搜索-b 参数指定的项以及它的所有子项。这是不带-s 时使用-b 的默认行为
-S attr	按指定的属性排序结果
-t	把二进制值写到临时目录下的文件中
-tt	把所有值写到临时目录下的文件中
-T path	写文件到 path 路径下，而不是临时目录
-u	输出对用户友好的显示名
-z limit	指定返回项的最大数目。如果没有指定此参数或指定的限制为 0，那么返回的项没有数量限制。但是，ldapsearch 返回的项绝不会多于服务器允许的数量

表 6-9　搜索过滤条件

运　算　符	用　途	样　例
=	查找所包含的属性值与指定值相同的项	“cn=John Browning”
= <string>* <string>	查找所包含的属性值与指定的子字符串相同的项	“cn=John*”“cn=J*Brown”
>=	查找特定项，该项中包含的属性的数字或字母值大于或等于指定的值	“cn>=D”
<=	查找特定项，该项中包含的属性的数字或字母值小于或等于指定的值	“roomNumber<=300″
=*	查找包含特定属性的值的项，而不用管属性的值是什么	“sn=*”
~=	查找特定项，该项中所含属性的值约等于指定的值	“sn~=Brning” 可能返回 sn=Browning
&	查找与所有搜索过滤器中指定的条件相匹配的项	“(&(cn=John Browning)(l=Dallas))”
|	查找与至少一个搜索过滤器中指定的条件相匹配的项	“(|(cn=John Browning)(l=Dallas))”
!	查找与任何搜索过滤器中指定的条件都不匹配的项	“(!(cn=John Browning)(l=Dallas))”

表 6-10　在查询过滤条件中必须进行转义的字符

字　符	十 进 制 值	十六进制值	转　义　串
*	42	0x2A	\2A
(	40	0x28	\28
)	41	0x29	\29
\	92	0x5c	\5C
NUL （null 字节）	0	0x00	\00

为了更好地说明 ldapsearch 命令的使用方法，下面给出一些实例。

例子 1：查找 ldap.example.com 上的所有目录项，并返回所有属性和值。

ldapsearch -h ldap.example.com “objectClass=*”

例子 2：查找 ldap.example.com 上的所有项，并返回 mail、cn、sn 等属性。

ldapsearch -hldap.example.com“objectClass=*” mail cn sn

例子 3：查找 ldap.example.com 上的“ou=West，o=Acme，c=US”条目下 cn 值以 Mike 开始的所有条目，并返回所有属性和值。

ldapsearch -b “ou=West, o=Acme, c=US” -h ldap.example.com “(cn=Mike*)”

例子 4：查找 ldap.example.com 上的所有条目并返回所有的属性和值，搜索时间限制为 5 秒。

ldapsearch -l 5 -h ldap.example.com“objectClass=*”

例子 5：返回 ldap.example.com 上的单条条目，把搜索基对象设为要返回条目的 DN，然后使用搜索范围为 base 的参数。

ldapsearch -h ldap.example.com -s base -b "uid=bjensen, ou=people, dc=example, dc=com" "(objectclass=*)"

例子 6：绑定用户“cn=John Doe, ou=people, dc=example, dc=com”，口令为明文“password”（此用户有查询权限），查找 ldap.example.com 上的所有条目，并以 LDIF 格式返回所有属性和值。

ldapsearch -h ldap.example.com -D “cn=john doe, ou=people, dc=example, dc=com” -w password -L “objectClass=*”

例子 7：组合过滤条件查询，如查找 ldap.example.com 上的属性为 chaoyang 或 haidian 的条目，返回所有属性。

ldapsearch -h ldap.example.com -s sub -b "dc=example, dc=com" "(|(L=chaoyang)(L=haidian))"

6.4.2　应用接口编程与实例

《RFC 1823 LDAP 应用程序接口》定义了 LDAP 的 C API 接口。下文根据 RFC1823 进行说明，其他语言的 API 是相似的。

应用程序通常按以下 4 个步骤使用 LDAP API。

① 打开一个到 LDAP 服务器的连接，ldap_open 返回连接句柄，允许多个连接同时打开。

② LDAP 服务器验证用户身份。ldap_bind 及其他相关函数支持不同的认证方法。

③ 执行 LDAP 操作获取结果。ldap_search 及相关函数返回的结果可以由 ldap_result2error、ldap_first_entry、ldap_next_entry 解析。

④ 关闭连接。调用 ldap_unbind 实现。

操作能够同步或异步执行。同步调用以_s 结尾，所以同步搜索的函数为 ldap_search_s。所有同步程序返回一个代表操作结果的指示符。（例如，常量 LDAP_SUCCESS 或其他错误码。）异步程序返回操作初始化的消息 ID，此 ID 可以被随后的 ldap_result 调用取得操作结果集。一个异步操作可以被 ldap_abandon 函数取消。

1. 调用 LDAP 操作

所有调用使用一个“连接句柄”（一个指向 LDAP 结构的指针），此结构包含每一个连接的信息，许多函数都返回 LDAPMessage 结构。

（1）打开一个连接

ldap_open 打开一个到 LDAP 服务器的连接。

```
typedef struct ldap {
        /* ... opaque parameters ... */
        int     ld_deref;
        int     ld_timelimit;
        int     ld_sizelimit;
        int     ld_errno;
        char    *ld_matched;
        char    *ld_error;
        /* ... opaque parameters ... */
} LDAP;
LDAP* ldap_open(char* hostname, int portno);
```

参数：

hostname：要连接的运行 LDAP 服务器地址，以空格分隔的主机名或 IP 地址列表。客户端依次尝试连接这些主机名，直到有一个连接成功。

portno：要连接的 TCP 端口号。

成功时，ldap_open 返回一个“连接句柄”，如果不能打开连接则返回 NULL。

（2）目录验证

ldap_bind 和相关函数用来进行目录验证。

```
int ldap_simple_bind( LDAP *ld, char *dn, char *passwd );
int ldap_simple_bind_s( LDAP *ld, char *dn, char *passwd );
int ldap_sasl_bind(LDAP *ld, const char *dn, const char *mechanism,
        const struct berval *cred, LDAPControl **serverctrls,
        LDAPControl **clientctrls, int *msgidp);
int ldap_sasl_bind_s(LDAP *ld, const char *dn, const char *mechanism,
        const struct berval *cred, LDAPControl **serverctrls,
        LDAPControl **clientctrls, struct berval **servercredp);
```

参数：

ld：连接句柄。

dn：绑定的条目名称。

passwd：ldap_simple_bind 使用的密码，如果为 NULL，则传递长度为 0 的密码到服务端。

mechanism：传递 LDAP_SASL_SIMPLE_NULL 以使用简单认证，或标识 SASL 方法的字符串。

cred：被验证的身份，mechanism 会确定 cred 的数据格式。

serverctrls：服务器端控制列表，如果为 NULL，表示没有服务器端控制。

clientctrls：客户器端控制列表，如果为 NULL，表示没有使用客户端控制。

msgidp：存储消息标识，用于获取操作结果。

servercredp：返回的服务器端身份标识、出错或无服务器端标识，设置为 NULL。

成功时，返回 LDAP_SUCCESS。

（3）关闭连接

ldap_unbind 用来与一个目录解除绑定并关闭连接。

Int ldap_unbind(LDAP* ld);

参数：

ld：连接句柄。

ldap_unbind 同步状态工作，同目录解除绑定，关闭连接，在返回前释放 ld 结构空间。ldap_unbind 返回 LDAP_SUCCESS（或其他请求不能送到 LDAP Server 的 LDAP 错误码）。调用 ldap_unbind 后，ld 连接句柄将不可用。

（4）查询

ldap_search 及其相关函数用来对 LDAP 目录进行查询，返回请求的每一个匹配条目的属性集。下面是三个相关函数：

```
struct timeval {
        long    tv_sec;
        long    tv_usec;
};
int ldap_search( LDAP   *ld, char *base, int   scope, char   *filter, char *attrs[], int attrsonly );
int ldap_search_s( LDAP *ld, char *base, int   scope, char *filter, char *attrs[], int   attrsonly,
                LDAPMessage   **res );
int ldap_search_st( LDAP *ld, char *base, int   scope, char *filter, char *attrs[], int   attrsonly, struct
                timeval   *timeout, LDAPMessage   **res   );
```

参数：

ld：连接句柄。ld 连接句柄的 3 个字段控制查询如何执行，它们是 ld_sizelimit、ld_timelimit、ld_deref。ld_sizelimit：限制查询返回的条目数量，0 代表无限制。ld_timelimit：限制查询时间，以秒为单位，0 代表无限制。ld_deref：常量 LDAP_DERF_NEVER、LDAP_DEREF_SEARCHING、LDAP_DEREF_FINDING、LDAP_DEREF_ALWAYS 之一，描述了在查询过程中如何处理别名。LDAP_DEREF_SEARCHING 意为在查询中别名被解除，但不会在定位于查询的基对象时解除引用。LDAP_DERED_FINGINFG 意为别名在定位基对

象但不是在查询过程中解除引用。异步查询由 ldap_search 初始化。函数返回查询初始化消息 ID。查询结果将被随后的 ldap_result 调用取得。结果由结果解释程序解释，在下面将详细介绍。如果出错，返回-1，同时 ld 连接句柄的 ld_errno 字段将会被相应设置。

base：开始搜索的 dn 条目。

scope：常量 LDAP_SCOPE_BASE、LDAP_SCOPE_ONELEVEL、LDAP_SCOPE_SUBTREE 之一，表示搜寻范围。

filter：RFC 1558 定义的字符串，表示搜索条件。

attr：指明要返回的属性，NULL 串将返回所有可用的属性。

attronly：布尔值，为 0 时指明返回属性的类型和属性值，非 0 值只返回所需类型。

timeout：在调用 ldap_search_st 时，定义本地查询的超时时间。

res：在同步调用时使用，作为返回结果参数，包含查询调用完成的结果。

同步查询通过调用 ldap_search_s 和 ldap_search_st 执行。这些函数除了 ldap_search_st 多一个附加参数以定义查询超时外，都是相同的。两个函数返回查询结果的标识、LDAP_SUCCESS 或者错误标识。查询条目返回结果包含在 res 参数中。此参数对于调用者是非透明的，条目、属性、值等应被后面说明的函数解析出来，包含在 res 中的结果应调用 ldap_msgfree 释放空间。

（5）读一个条目

LDAP 不支持直接的读操作，代之以查询来模拟，设置 base 参数为要读的条目 DN，scope 设置为 LDAP_SCOPE_BASE，filter 设置为“(objectclass=*)”。attrs 包含返回的属性列表。

（6）列出一个条目的子条目

LDAP 不支持直接列表操作，代之以查询来模拟，设置 base 为要列表的条目 DN，scope 设为 LDAP_SCOPE_ONELEVEL，filter 设为“(objectclass=*)”。attrs 包含返回的每一个子条目的属性列表。

（7）修改条目

ldap_modify 和 ldap_modify_s 函数用来修改已存在的 LDAP 条目。

```
typedef struct ldapmod {
    int    mod_op;
    char *mod_type;
    union {
       char **modv_strvals;
       struct berval     **modv_bvals;
    } mod_vals;
} LDAPMod;
#define mod_values          mod_vals.modv_strvals
#define mod_bvalues         mod_vals.modv_bvals
int ldap_modify( LDAP *ld, char *dn, LDAPMod *mods[] );
int ldap_modify_s( LDAP *ld, char *dn, LDAPMod *mods[] );
```