多个用户。举例来说，DN 为 uid=matt, ou=Users, dc=example, dc=com 就是 ou=Users, dc=example, dc=com 的子树，当试图要访问它时，这个 ACL 指令就起了作用。

总体的意思是，任何人都没有权限访问 ou=Users, dc=example, dc=com 及其子树的信息。

DN 表示方式分别是：

① dn.base：约束这个特定 DN 的访问。它和 dn.exact、dn.baselevel 是相同的意思。

② dn.one：约束这个特定的 DN 第一级子树的访问。它和 dn.onelevel 是同义词。

③ dn.children：和 dn.subtree 类似，都是对其以下的子树访问权的约束。不同点在于，这个约束是不包含自己本身的 DN，而 subtree 包含了本身的 DN。

对于 dn 的约束条件还可以利用正则表达式，如下：

```
access to dn.regex="uid=[^, ]+, ou=Users, dc=example, dc=com" by * none
```

这个指令将约束所有 uid=(任何值), ou=Users, dc=example, dc=com 的 DN，其中的任何值是用[^,]+这个符号组合来表示的，可以代表任何至少有 1 个字符且字符当中没有逗号(,)的值。更明确点说，就是在 ou=Users, dc=example, dc=com 这个 DN 下所有以 uid 为属性的一级子树都属于这个约束的范围。

（2）通过约束 attrs 访问

对于 DN 的约束大多用在对某个层级的约束，而使用 attrs 就可以跨层级（或者跨越父类树），通过属性来约束访问的范围。

```
access to attrs=homePhone by * none
```

这个例子的意思是：任何人都没有权限访问属性为 homePhone 的信息。

在 attrs 后面的值可以有多个，例如：

```
access to attrs=homePhone, homePostalAddress
```

如果要约束某个对象类的所有属性，可以采用以下形式：

```
access to attrs = title, registeredAddress, destinationIndicator, …
```

但这个方法太耗时，也难以阅读，显得笨重，以下给出一个好的方法：

```
access to attrs=@organizationalPerson by * none
```

采用@的方法必须谨慎，这段指令不仅仅约束了 organizationalPerson 里的属性，也约束了 person 对象类的属性。为什么？因为 organizationalPerson 对象类是 person 的子类，因此，所有 person 中的属性当然也是 organizationalPerson 的属性。

如果想做除了 organizationalPerson 的其他对象类的约束，可以用“!”来表示：

```
access to attrs=!organizationalPerson
```

也可以加入属性的值，具体约束某个值：

```
access to attrs=givenName val="Matt"
```

这个指令也可以采用正则表达式约束的方法，如下：

```
access to attrs=givenName val.regex="M.*"
```

最后给出一个一般情况下用到的利用属性约束的例子：

```
access to attrs=member val.children="ou=Users, dc=example, dc=com"    by * none
```

（3）通过过滤（filter）访问

过滤提供一种支持条目记录匹配的方法，如下：

```
access to filter="(objectClass=simpleSecurityObject)" by * none
```

这表示可以约束所有记录中包含对象类为 simpleSecurityObject 的信息。

与编程语言类似，ACL 指令也有类似与或的条件判断，如下：

```
access to filter="(|(|(givenName=Matt)(givenName=Barbara))(sn=Kant))" by * none
```

这段代码过滤出 givenName 为 Matt 或 Barbara 或者 surname 为 Kant 的信息。

6.1.7 LDIF 数据交换文件

LDAP 数据交换格式（LDAP Data Interchange Format）是在 RFC2849 中定义的基于文本描述目录条目的一种标准格式，即使两个服务器使用不同的内部数据存储格式，LDIF 也可以导出目录数据并将其导入到另一个目录服务器。

LDIF 文件是文本文件，可以由 5 种类型的行构成：指令行、续行、空行、注释行、分隔行。空行就是不包含任何字符的行；注释行是以井号（#）开头的行；分隔行是只有减号（-）的行，它用来分隔对一个目录条目的多个操作；续行是以一个空格开头的行；指令行是除了以#号和空格开头的行。

有两种不同类型的 LDIF 文件。第一类描述了一组目录条目，如整个企业目录，或者是企业目录的一个子集；另一种类型的 LDIF 文件是一系列的目录条目更新语句，用于更新现有的目录条目数据，在 RFC 2849 中有完整格式的定义。下面分别说明。

6.1.7.1 第一种类型文件

第一种类型文件的内容包含两部分：第一部分是 DN，第二部分是一系列的属性值对。如下所示：

```
dn: uid=bjensen, ou=people, dc=example, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Barbara Jensen
cn: Babs Jensen
givenName: Barbara
sn: Jensen
uid: bjensen
mail: bjensen@example.com
telephoneNumber: +1 408 555 1212
description: Manager, Switching Products Division
```

DN 必须是条目的第一行，由字符串“DN”后跟一个冒号（:）和条目的区别名组成。DN 后面是条目的属性，每个属性由一个属性类型、一个冒号（:）和属性值组成。属性可以以任何顺序出现，以增强可读性。但是一般先列出条目的对象类，并把相同属性类型的值放在一起。

当一行数据很长时，通常的做法是对数据进行换行。LDIF 文件支持换行，具体做法是：在折行的地方插入一个换行符和空格符。如下所示：

```
description: I will be out of the
 office from August 12, 2001, to September 10, 2001.
```

如果 LDIF 中的属性值不为 ASCII，则必须用 Base64（参考）进行编码，采用 Base64 编码的值用两个冒号（::）分隔属性和值，如：

```
jpegPhoto:: /9j/4AAQSkZJRgABAAAAAQABAAD/2wBDABALDA4MChAODQ4
 SERATGCgaGBYWGDEjJR0oOjM9PDkzODdASFxOQERXRTc4UG1RV19iZ2hnP
```

6.1.7.2　第二种类型文件

第二种类型文件包含更新语句。第一行同样是 DN，第二行是更新类型，后面是要更新的属性及值，也可以用来添加新的条目。

1．增加条目

changetype 类型 add 表示增加一个条目到目录中，格式为：

```
dn: 要增加条目的 DN 值
changetype: add
attribute type: value
```

2．删除条目

changetype 类型 delete 表示从目录中删除一个条目，格式为：

```
dn: 要删除条目的 DN 值
changetype: delete
```

3．修改条目

changetype 类型 modify 表示修改目录中的一个条目，可以增加新的属性值，删除指定属性值，删除全部属性，替换属性值为新值，格式为：

```
dn: 要修改条目的 DN 值
changetype: modify
modifytype: attribute type
[attribute type: attribute value]
```

可以看到上面增加了新的操作符 modifytype，它的值可为 add、delete 或 replace，示例如下：

① 增加 telephoneNumber 属性：

```
dn: uid=bjensen, ou=people, dc=example, dc=com
```

```
changetype: modify
add: telephoneNumber
telephoneNumber: +1 216 555 1212
telephoneNumber: +1 408 555 1212
```

② 删除 telephoneNumber 属性的+1 216 555 1212 值：

```
dn: uid=bjensen, ou=people, dc=example, dc=com
changetype: modify
delete: telephoneNumber
telephoneNumber: +1 216 555 1212
```

③ 完全删除 telephoneNumber 属性值：

```
dn: uid=bjensen, ou=people, dc=example, dc=com
changetype: modify
delete: telephoneNumber
```

④ 替换 telephoneNumber 属性值为 2 项新值：

```
dn: uid=bjensen, ou=people, dc=example, dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 216 555 1212
telephoneNumber: +1 405 555 1212
```

⑤ 多个操作可以组合起来使用，中间用单独一行“-”进行分隔，如：

```
dn: uid=bjensen, ou=people, dc=example, dc=com
changetype: modify
add: mail
mail: bjensen@example.com
-
delete: telephoneNumber
telephoneNumber: +1 216 555 1212
-
delete: description
-
replace: givenName
givenName: Barbara
givenName: Babs
-
```

4. 重命名或移动条目

重命名或移动条目可使用值为 moddn 的 changetype 操作，修改了 DN，可把条目移动到目录树的新位置。

```
dn: 要修改条目的 DN 值
changetype: moddn
[newsuperior: 新的父条目 DN]
[deleteoldrdn: ( 0 | 1 )]
[newrdn: 条目新 RDN]
```

如果修改 RDN，则必须提供 newrdn 和 deleteoldrdn 参数；如果把条目移动到新位置，则必须提供 newsuperior 参数。修改 RDN 示例如下：

```
dn: uid=bjensen, ou=People, dc=example, dc=com
changetype: moddn
newrdn: uid=babsj
deleteoldrdn: 0
```

移动条目到新位置示例：

```
dn: uid=bjensen, ou=People, dc=example, dc=com
changetype: moddn
newsuperior: ou=Terminated Employees, dc=example, dc=com
```

6.2　常见 LDAP 产品介绍

6.2.1　IBM TDS

IBM Tivoli 目录服务器（ITDS）的前身为 IBM 目录服务器，实现了轻量级目录访问协议 LDAP，是 IBM 的 Tivoli 身份与访问管理产品的一部分。ITDS 可以跨平台进行安装配置。

ITDS 提供了以下组件：一个使用 DB2 数据库对目录信息进行存储的服务器，一个将 LDAP 操作路由到其他服务器上的代理服务器，一个客户端，一个管理服务器的图形界面，一个管理用户的图形界面。

ITDS 支持多种身份验证方式，包括：用户名和密码验证、数字证书身份验证、简单验证和安全层（SASL）、挑战-响应身份验证机制（CRAM-MD5）、Kerberos 身份验证。

ITDS 具有如下特点：

① 支持数百万目录条目。使用 IBM DB2 技术存储目录数据。
② 实现 LDAP 相关规范。提供符合 LDAP 规范的随需应变的身份基础设施。
③ 利用强大的主/从和对等复制提供高可用性，支持多达数十个主服务器。
④ 集成了 IBM 中间件、身份管理和安全产品，支持与非 IBM 产品集成。
⑤ 支持基于 Web 进行系统管理。
⑥ 支持主流平台，包括 AIX、Solaris、Windows Server、HP-UX、SUSE、Red Hat。

6.2.2　Sun Java 系统目录服务器

Sun Java 系统目录服务器（Sun Java System Directory Server）早期称为 Sun ONE 目录服务器、iPlanet 目录服务器，更早之前称为 Netscape 目录服务器。它包括 Sun LDAP 目录服务器和 DSML 服务器，是 Java 企业系统的一个组件。