OpenLDAP 2.1 管理员指南

The OpenLDAP Project <http://www.openldap.org/>

9 June 2002

该中文文档由 NTKO 翻译

E-mail: dominoreg@sina.com

Any comments are welcome!

This document is form OpenLDAP 2.1 Administrator's Guide English Version

由于时间关系，质量一般。尤其是5.3章！

欢迎指正。

目录

0、 前言

0.1、 版权

0.2、 该文档的范围

该文档提供了为在UNIX或者类UNIX的系统上安装OpenLDAP 2.1软件（http://www.openldap.org/software/）提供了一个指南。该文档面向有经验的系统管理员。但是，他可以没有运行一个基于LDAP目录软件的经验。

该文档应该和随该软件包一起提供的其他的OpenLDAP信息资源，以及和WWW上该项目的站点（http://www.OpenLDAP.org/）上提供的信息一起使用。该站点提供了大量的资源。

OpenLDAP Resources	
Resource	**URL**
Document Catalog	http://www.OpenLDAP.org/doc/
Frequently Asked Questions	http://www.OpenLDAP.org/faq/
Issue Tracking System	http://www.OpenLDAP.org/its/
Mailing Lists	http://www.OpenLDAP.org/lists/
Software Pages	http://www.OpenLDAP.org/software/
Support Pages	http://www.OpenLDAP.org/support

0.3、 致谢

OpenLDAP Project由一个志愿者小组组成。没有他们贡献的时间和精力，该文档不可能形成。

OpenLDAP项目同时还感谢University of Michigan LDAP，他们创建了LDAP软件的基础。并且

OpenLDAP软件就是在他们的软件之上创建的。该文档也是在U-Mich的LDAP文档：*The SLAPD and SLURPD Administrators Guide*的基础上形成的。

0.4、 错误修正

任何对于该文档的改正和完善建议应该使用OpenLDAP 的*Issue Tracking System* (http://www.openldap.org/its/)提交。

0.5、 关于该文档

该文档使用由*Ian Clatworthy*开发的简单文档格式（*Simple Document Format*）文档系统（http://www.mincom.com/mtr/sdf/）创建。

1、 介绍 OpenLDAP 目录服务

该文档描述了如何编译，配置和运行OpenLDAP软件来提供目录服务。包含了如何配置和运行独立的LDAP daemon，slapd(8)，以及如何配置运行独立的LDAP更新复制daemon，slurpd(8)的详细信息。该文档面向新手和有经验的系统管理员。本部分提供了对目录服务的一个基本的介绍，特别是对slapd(8)提供的目录服务。

1.1、 什么是目录服务？

一个目录服务是一个特殊的数据库，它为读，浏览和搜索进行了优化。目录可以用来保存描述性的，基于属性的信息，并且支持复杂的过滤搜索能力。目录通常不支持在一般的数据库管理系统中支持的复杂的事务处理或者回滚机制——这些机制是为处理大容量的复杂更新操作而设计的。目录更新只是简单的“所有或者没有”的更新——如果它们被允许的话。目录被优化为针对大量的查找或者搜索操作进行快速的响应。它们可以具有大范围复制信息的功能，以便提高可用性和可靠性，同时缩短响应时间。

当目录中的信息被复制的时候，复本之间信息的暂时不一致是允许的，只要它们最终达到了一致。

有许多种不同的方法来提供一个目录服务。不同的方法允许在目录中存储不同种类的信息，因此，信息如何被引用，查询和更新的要求也就不同，信息被保护并拒绝未经授权访问的方式等等也就不同。一些目录服务是本地的，提供有限制的服务（比如，一个运行在单一机器上的finger服务）。还有一些服务是全局的，提供大的多的规模的服务（比如，整个Internet）。全局服务通常是分部的。这意味着他们保存的数据在许多台机器上分部，所有这些机器合作提供目录服务。典型情况下，一个全局服务定义了一个统一的名字空间，无论您从何处开始和数据关联，该名字空间都会提供相同的数据视图。Internet域名系统就是一个全局分布的目录系统服务的实例。

1.2、什么是LDAP？

LDAP代表轻型目录访问协议。正如名字中表示的，它是一个用来访问目录服务，尤其是基于X.500的目录服务的轻型协议。LDAP运行在TCP/IP协议或者其他的面向连接的传输服务之上。LDAP的详细本质在RFC2251“轻型目录访问协议(V3)”中定义。本部分从用户的角度给出了LDAP的概述。

什么类型的信息可以被保存在目录中？LDAP的信息模型是建立在“条目”（entries）的基础上。一个条目是一个属性的集合，并且具有一个全局唯一的“可区分名称”DN。该DN被用来唯一的引用该条目。每一个条目的属性具有一个类型和一个或者多个值。类型通常是容易记忆的名称，比如“cn”是通用名称（common name），或者“mail”是电子邮件地址。条目的值的语法取决于属性类型。比如，cn属性可能具有一个值“Babs Jensen”。一个mail属性可能包含值“babs@example.com”。一个jpegphoto属性可能包含一幅JPEG（二进制）格式的图片。

信息是怎样组织的？在LDAP中，目录条目以一个层次的树结构来安排。传统上，该结构反映了地理的或者组织结构上的边界。条目代表国家，出现在树的顶部，下面是代表州或者地理组织的条目，它们下面可能是代表组织单元，个人，打印机，文档或者只是任何其他的你能想到的东西的条目。图1.1显示了一个使用传统命名的LDAP目录树的例子：

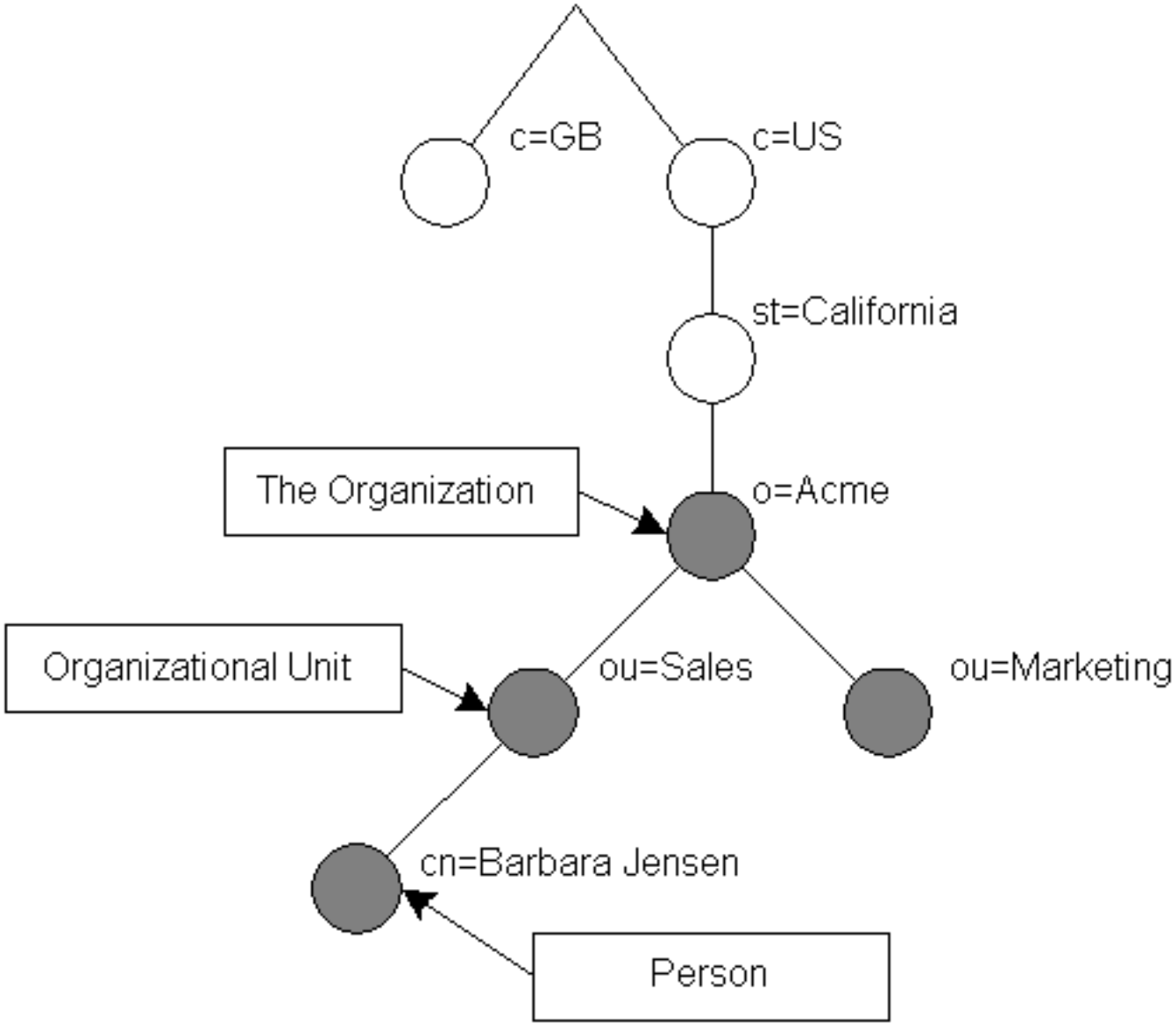

图**1.1: LDAP**目录树（传统命名方式）

该树同样可以根据Internet域名来安排。这种命名方式逐渐普遍，因为它允许目录服务使用域名系统来定位。图1.2显示了一个使用基于域名的命名方式的LDAP目录树的示例。

另外，LDAP通过使用一个特殊的，叫做objectClass的属性来允许您控制哪一个属性必须出现或者允许出现在一个条目中。objectClass属性的值决定了该条目必须遵守的模式规则。

信息是怎样被引用的？一个条目通过它的DN被引用。该DN使用如下方式构造：首先得到条目自己的名称（相对DN，RDN），然后连接上其祖先条目的名称。比如，图1.2的基于Internet的命名方式中的Barbara Jensen的条目，具有一个RDN：uid=babs。和一个DN：uid=babs,ou=People, dc=example,dc=com。完全的DN格式在RFC2253“ Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names ”中描述。

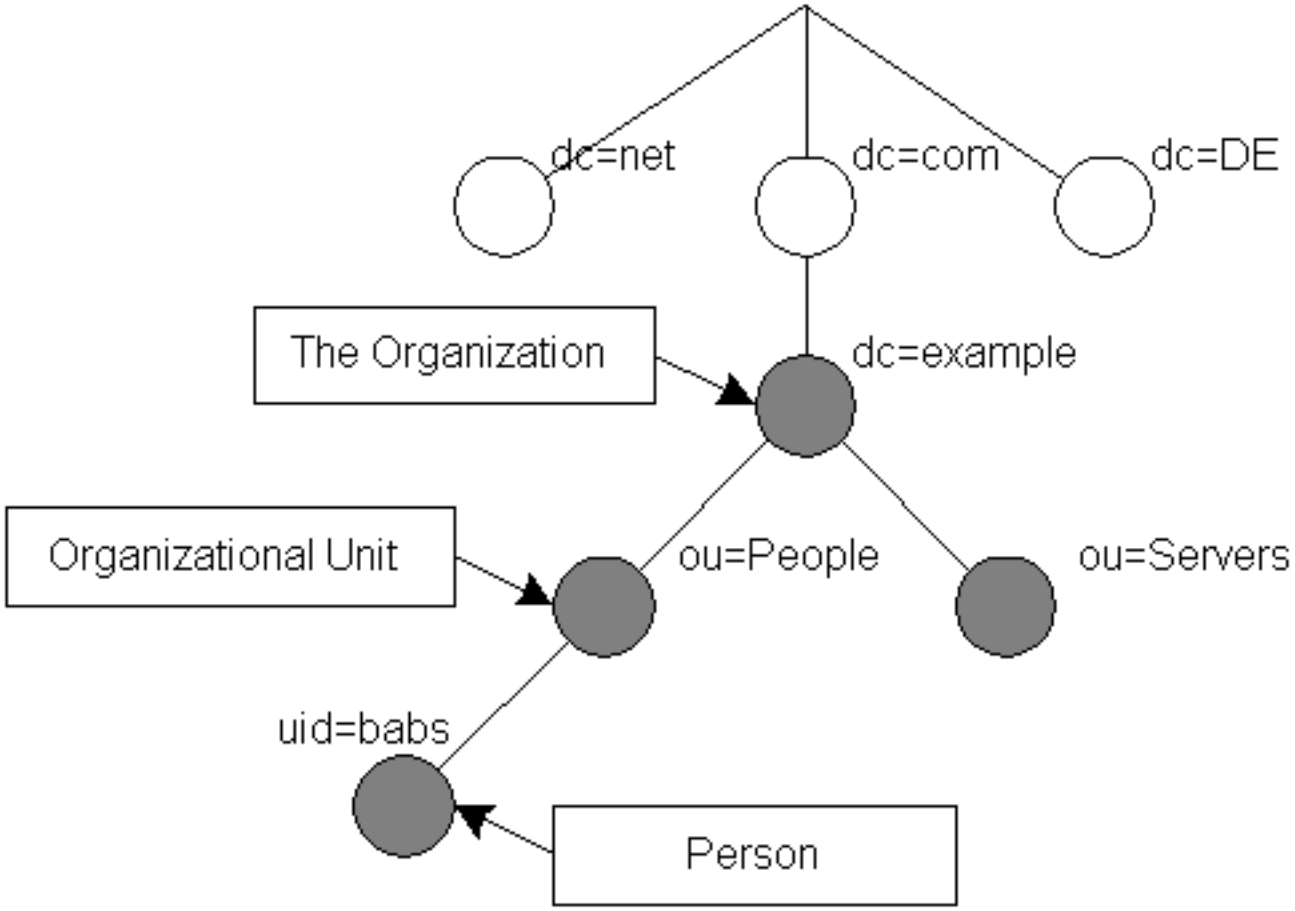

图**1.2: LDAP**目录树（基于**Internet**的命名）

信息是如何被访问的？LDAP定义了查询和更新目录的操作。还提供了从目录中增加和删除一个条目的操作，以及改变条目名称的操作——虽然大多数时间LDAP被用来在目录中搜索信息。LDAP搜索操作允许搜索目录的一部分，查找匹配由某个搜索过滤器指明的规则的条目。然后就可以从匹配规则的每一个条目中获取信息。

比如，您需要搜索在dc=example,dc=com条目之中或者之下的整个目录子树，查找一个名字叫做Barbara Jensen的个人，并且获取每一个找到的条目的电子邮件地址。LDAP让您简单的做到这一点。或者，您可能需要搜索直接在st=California,c=US条目之下的条目，查找一个组织名字中包含字符串Acme，并且具有传真号码的组织。LDAP也允许您做到这一点。下一部分将更详细的描述您可以使用LDAP做什么，以及为什么它对您很有用。

信息是怎样被保护以拒绝未经授权的访问？一些目录服务提供了未经保护的信息，允许任何人看到这些信息。LDAP提供了一种机制，让客户端进行认证，或者向目录服务器证实它的身份，这为保护服务器包含的信息提供丰富的存取控制铺平了道路。LDAP同时还支持机密性，完整性和安全性的服务。

1.3、 LDAP 如何工作？

LDAP目录基于客户/服务器方式实现。一个或者多个LDAP服务器包含了组成目录信息树（DIT）的数据。客户连接到服务器并提问，服务器返回答案或者一个指向其他信息的指针（通常是另外一个LDAP服务器）无论客户首先连接到哪一个LDAP服务器，它看到相同的目录视图。提交到一个LDAP服务器的名字和另外一个LDAP服务器将引用了相同的条目。这是一个全局目录服务，比如LDAP的一个重要特性。

1.4、 关于 X.500

从技术上说，LDAP是一个访问一个X.500目录服务——OSI目录服务的目录访问协议。最初，LDAP客户端通过网关访问目录服务。该网关运行LDAP（在客户和网关之间）和X.500的目录访问协议（DAP）（在网关和X.500服务器之间）。DAP是一个重量级的协议，运行在去安全的OSI协议栈之上，需要消耗大量的计算资源。LDAP被设计出来运行在TCP/IP上，并且以低的多的代价提供了大多数DAP所能提供的功能。

虽然LDAP仍然可以被用来通过网关来访问X.500目录服务，现在，LDAP更多的被直接在X.500服务器上实现。

独立的LDAP daemon，或者说slapd(8)，可以被看作是一个轻型的X.500目录服务器。也就是说，它没有实现X.500的DAP，作为一个轻型的目录服务器，slapd(8)只是实现了X.500模型的一个功能子集。

如果您已经运行了一个X.500 DAP服务器，并且您想继续这样做，您可以停止阅读本指南。该指南都是关于通过slapd(8)来运行LDAP的，没有运行X.500 DAP。如果您没有运行X.500 DAP，或者想停止运行X.500 DAP，或者没有马上运行X.500 DAP的计划，请继续阅读。

从一个LDAP目录服务器向一个X.500 DAP DSA复制数据是可行的。这需要一个LDAP/DAP网关。OpenLDAP没有提供这样一个网关。但是，我们的复制daemon可以被用来向这样的网关进行复制。参阅本文档的“ 使用SLURPD进行复制 ”一章获得有关复制的信息。

1.5、 LDAP V2 和 V3 的不同

LDAPV3增加了下面的特性：

- 使用SASL进行强度认证；
- 使用TLS(SSL)进行完整性和安全保护；
- 使用UNICODE支持国际化；
- 引用和持续

¢ 可扩展性（控制和扩展操作）

¢ 支持模式发现

同时支持LDAPV2和V3是有问题的，应该尽量避免。因为LDAPV3包含了LDAPV2的所有特性，因此，推荐使用LDAPV3；

1.6、 SLAPD 的介绍和作用

SLAPD(8)是一个跨平台的LDAP服务器。可以使用它来提供自己的目录服务。自己的目录可以包含任意多的信息。可以将它和一个全局的LDAP目录服务相联接，或者只是自己运行一个服务。下面是SLAPD的更多特性：

1.6.1、 LDAPV3：

SLAPD实现了LDAP协议的V3版本。并且支持IPV4和IPV6上的LDAP。

1.6.2、 简单认证和安全层

SLAPD通过使用SASL来支持强度认证服务。SLAPD的SASL实现使用了Cyrus SASL软件，该软件支持大量的认证机制，包括DIGEST-MD5，EXTERNAL和GSSAPI。

1.6.3、 传输层安全

SLAPD通过使用TLS（或者SSL）来提供完整性和安全保护。SLAPD的TLS实现使用了OpenSSL软件。

1.6.4、 拓扑控制

SLAPD允许给予网络拓扑控制对服务器的访问。这个特性使用了TCP wrappers。

1.6.5、 存取控制

SLAPD提供了丰富和功能强大的存取控制手段，允许控制数据库的信息。可以基于LDAP的认证信息，IP地址，域名或者其他的条件来控制对条目的存取。SLAPD支持静态和动态的存取控制信息。

1.6.6、 国际化

SLAPD支持UNICODE和语言标记。

1.6.7、 可选择的后台数据库

SLAPD可以选择集中不同的数据库后台支持。包括BDB——高性能的事务数据库；LDBM，轻型的基于DBM的后台数据库；SHELL，多种shell脚本的后台界面；以及PASSWD，一个简单的passwd(5)的界面。BDB使用了Sleepycat的Berkeley DB 4。LDBM使用了Berkeley DB或者GDBM。

1.6.8、 多个数据库实例

SLAPD可以被配置为同时使用多个数据库。这意味着一个SLAPD服务器可以使用相同或者不同的后台数据库，响应多个逻辑上不同的LDAP部分树上的请求。

1.6.9、 通用的模块 API

如果您需要更多的定制，SLAPD允许很容易的书写自己的模块。SLAPD包含了2个清楚的部分：一个前端来处理和LDAP客户之间的协议通信；以及一些处理特定任务比如数据库操作的模块。因为这两个部分之间通过定义的非常好的C API进行通信，您可以书写自己定制的模块，使用多种方法扩展SLAPD。同时，SLAPD还提供了多个可编程的数据库模块。这些允许您使用通用的编程语言（PERL，SHELL，SQL和TCL）将外部数据源开放给SLAPD。

1.6.10、 线程

SLAPD使用了线程来达到高性能。一个单一的多线程SLAPD进程使用线程池来处理所有的请求。这减少了系统的负载，并且显示了高性能。

1.6.11、 复制

SLAPD可以被配置为保持其数据库的复制拷贝。这种单一主服务器/多个附属服务器的复制模式在高吞吐量的环境下是必须的。这时，一个单一的SLAPD不能提供必需的可用性和可靠性。SLAPD也包括了一个实验性的支持多个主服务器的复制。

1.6.12、 可配置性

SLAPD通过一个单一的配置文件来实现高可配置性。此配置文件允许您改变需要改变的任何东西。配置选项具有合理的缺省选项，这使得您的工作更加容易。

当然，SLAPD同时具有其限制。主要的后台BDB数据库不能很好的处理范围查询或者嵌套查询。

1.7、 SLURPD 及其作用

SLURPD(8)是一个服务器daemon，它帮助SLAPD提供复制服务。它的责任在于将对主SLAPD数据库的修改分发到不同的SLAPD的副本上。它使得SLAPD避免去考虑当改变发生的时候，一些副本可能DOWN机，或者不可访问。SLURPD自动重新尝试失败的请求。SLAPD和SLURPD通过一个简单的用来记录改变的文件进行通信。

有关如何配置和运行SLURPD的信息，请参考“ 使用SLURPD进行配置 ”一章。

2、 快速指南

下面的部分是OpenLDAP2.1的一个快速指南，包括独立的LDAP daemon，SLAPD(8)。

如果您需要认真地运行OpenLDAP，在安装之前应该阅读本文档的所有部分。

注意：快速指南没有使用强度认证和任何完整性和安全保护服务。这些服务在**OpenLDAP**管理员指南的后面的章节描述。

2.1、 获得软件

http://www.openldap.org/software/download/。

2.2、 解开软件发行包

```
gunzip _c openldap-VERSION.tgz | tar xvfB –
```

然后进入目录：

```
cd openldap-VERSION
```

您必须将VERSION使用发行版本的版本名称替换。

2.3、 查看文档

应该查看随版本发行的COPYRIGHT，LICENSE，README，以及INSTALL文档。COPYRIGHT和LICENSE提供了OpenLDAP软件可以使用，拷贝和限制的警告。

还应该查看本文档的其他章节。特别是，“ 编译和安装OpenLDAP软件 ”。这一章提供了该软件需要的其他软件和安装步骤地详细信息。

2.4、 运行 configure

您需要运行系统提供的configure脚本来配置您系统上的发行版本。configure脚本接受许多命令行参数，它们允许或者禁止软件的某些特性。通常情况下，默认的选项是可以的，但是，您可能需要改变它们。要得到configure脚本接受的命令行参数的列表，使用—help选项：

```
./configure –-help
```

但是，既然您使用了本指南，我们假设您足够勇敢，只是让configure来决定什么是最佳的：

```
./configure
```

假设configure喜欢您的系统，您可以继续编译软件。如果configure的确抱怨了什么，现在您需要查看FAQ的安装部分（http://www.openldap.org/faq/），或者，阅读本文档的“ 编译和安装OpenLDAP软件 ”部分。

2.5、 编译软件

下一个步骤是编译软件。这个步骤有两部分，首先，我们创建编译依赖选项，然后编译软件：

```
make depend
```

```
make
```

两次make都应该能够进行无错误编译。

2.6、 测试编译

为了确保进行了正确的编译，应该测试代码集合（它只需要几分钟时间）：

```
make test
```

符合您的配置的测试程序将会运行，它们应该通过。一些测试，比如复制测试，可以跳过。

2.7、 安装软件

现在可以准备安装软件了。这通常需要超级用户的权限：

```
su root –c ‘make install’
```

所有东西应该被安装到/usr/local（或者是configure中指定的安装目录）

2.8、 编辑配置文件

使用编辑器编辑系统提供的slapd.conf（5）示例（通常安装在/usr/local/etc/openldap/slapd.conf）来包含一个如下所示的BDB数据库定义：

```
database bdb

suffix “dc=<MY-DOMAIN>,dc=<COM>”

rootdn “cn=Manager,dc=<MY-DOMAIN>, dc=<COM>”

rootpw secret

directory /usr/local/var/openldap-data
```

确保记住使用正确的域名替换<MY-DOMAIN>和<COM>。比如，对于example.com，使用：

```
database bdb

suffix “dc=example,dc=com”

rootdn “cn=Manager,dc=example, dc=com”

rootpw secret

directory /usr/local/var/openldap-data
```

如果您的域名包含了其他的部分，比如，eng.uni.edu.eu，使用：

```
database bdb

suffix “dc=end,dc=uni,dc=edu,dc=eu”

rootdn “cn=Manager, dc=end,dc=uni,dc=edu,dc=eu”

rootpw secret

directory /usr/local/var/openldap-data
```

配置slapd(8)的详细信息可以在slapd.conf(5)的手册页中或者在本文档的“ SLAPD配置文件 ”一章

中找到。

注意：指定的目录在启动**slapd(8)**之前必须存在。

2.9、 启动SLAPD

现在已经准备好启动独立的LDAP服务器，slapd(8)了。运行下面的命令：

```
su root –c /usr/local/libexec/slapd
```

为了检查服务是否正在运行并且被正确配置，可以对服务器运行一个搜索命令，使用ldapsearch (1)。ldapsearch安装在/usr/local/bin/ldapsearch：

```
ldapsearch –x –b '' –s base '(objectclass=*)' namingContexts
```

注意命令行参数中单引号的使用，它们阻止特殊的字符被shell解析。这应该返回：

```
dn:

namingContexts: dc=example,dc=com
```

关于运行slapd(8)的详细信息，可以在slapd(8)的手册页或者本文档的“ 运行SLAPD ”章节中找到。

2.10、 向目录中增加初始化条目

可以使用ldapadd(1)来向LDAP目录中增加条目。ldapadd需要输入是LDIF格式。我们可以通过两个步骤来完成这一点：

- 创建一个LDIF文件；
- 执行ldapadd;

使用编辑器创建一个LDIF文件，它包含：

```
dn:dc=<MY-DOMAIN>,dc=<COM>

objectclass:dcObject

objectclass:organization
```

```
o:<MY ORGANIZATION>

dc:<NY-DOMAIN>

dn:cn=Manager,dc=<MY-DOMAIN>, dc=<COM>

objectclass:organizationalRole

cn:Manager
```

确保使用正确的域名部分替换<MY-DOMAIN>和<COM>。并且使用您组织的名称替换<MY ORGANIZATION>。如果您使用了拷贝粘贴，确认将开始和结束的空格从下面的例子中去掉。

```
dn:dc=example,dc=com

objectclass:dcObject

objectclass:organization

o:Example Company

dc:example

dn:cn=Manager, dc=example,dc=com

objectclass:organizationalRole

cn:Manager
```

现在，您可以运行ldapadd(8)来讲这些条目增加到目录中（您会被提示输入slapd.conf中指定的”secret”部分）：

```
ldapadd –x –D "cn=Manager,dc=example,dc=com" -W -f example.ldif
```

其中，example.ldif是上面您创建的文件。

其他的关于创建目录的信息可以在本文档的“数据库创建和维护工具”一章中找到。

2.11、是否能够工作？

现在来检验增加的条目确实在数据库中。可以使用任何LDAP客户端来验证这一点。但是，我们

的例子是用ldapsearch(1)工具。记住，使用您的正确的域名替换dc=example,dc=com：

```
ldapsearch -x -b 'dc=example,dc=com' '(objectclass=*)'
```

这个命令将搜索并且获取数据库中的所有条目。

现在可以使用ldapadd(1)或者其他的LDAP客户端，试验不同的配置选项和后台的配置等等。

注意默认情况下，slapd(8)数据库给除了super-user之外的每一个人读者的存取权限。强烈建议设置对存取权限的控制来认证用户。存取控制在“ SLAPD配置文件 ”一章中的“ 存取控制 ”部分讨论。同时，建议您阅读“ 使用SASL ”和“ 使用TLS ”部分。下面的部分提供了有关编译，安装和运行slapd(8)的更详细的信息。

3、 配置选项

下面的部分给出了不同的LDAP目录配置选项，以及您的独立的LDAP服务器slapd(8)如何适应其他的服务器。

3.1、 本地目录服务

在此配置中，您运行一个slapd只为您本地的域提供目录服务。它不会和其他的目录服务器交互。这种配置如图3.1所示：

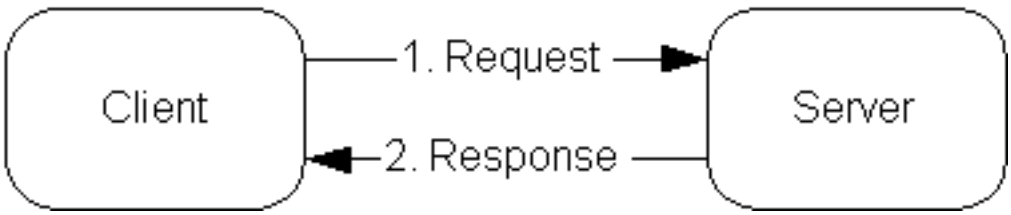

图**3.1:**本地目录服务配置

如果你只是刚刚开始，或者如果您需要提供本地的服务，并且对于连接到世界上的其他部分不感兴趣，请使用这个配置。（这是第2章快速指南中的配置）。如果需要，以后可以很容易的升级到其他的配置。

3.2、 带有引用（Referrals）的本地目录服务

在这个配置中，您运行一个slapd，它为您的本地域提供目录服务，并且将它配置为，对于本地域之外的请求，返回一个指向上一级目录的引用。您可以自己运行该服务，或者使用其他人为您提供的服务。这种配置如图3.2所示：

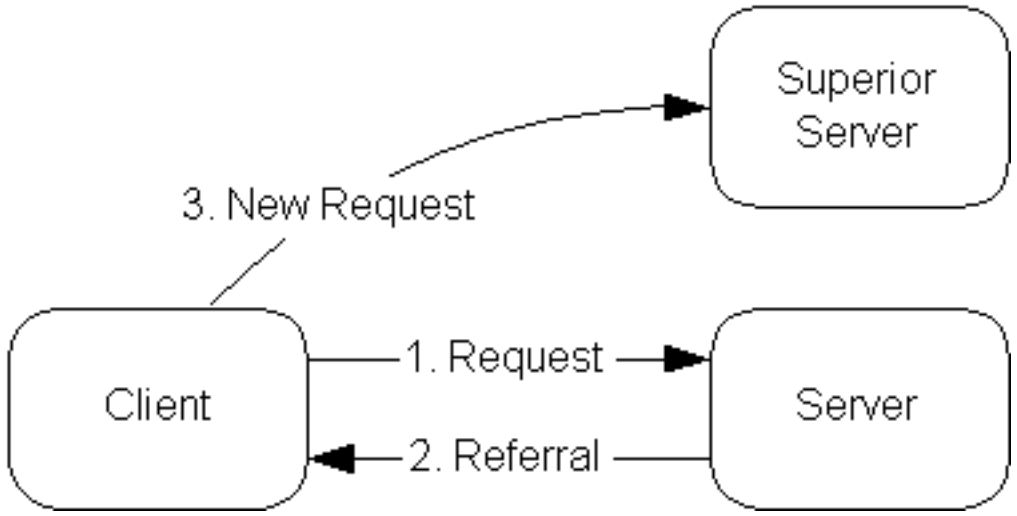

图**3.2:**带有引用的本地目录服务

如果您需要提供本地的服务并且参与到一个全局服务，使用此配置。

3.3、 可复制的目录服务

SLURPD daemon用来从主slapd服务器向一个或者多个附属slapd服务器传播改变。一个主——附属配置的例子如图3.3所示。

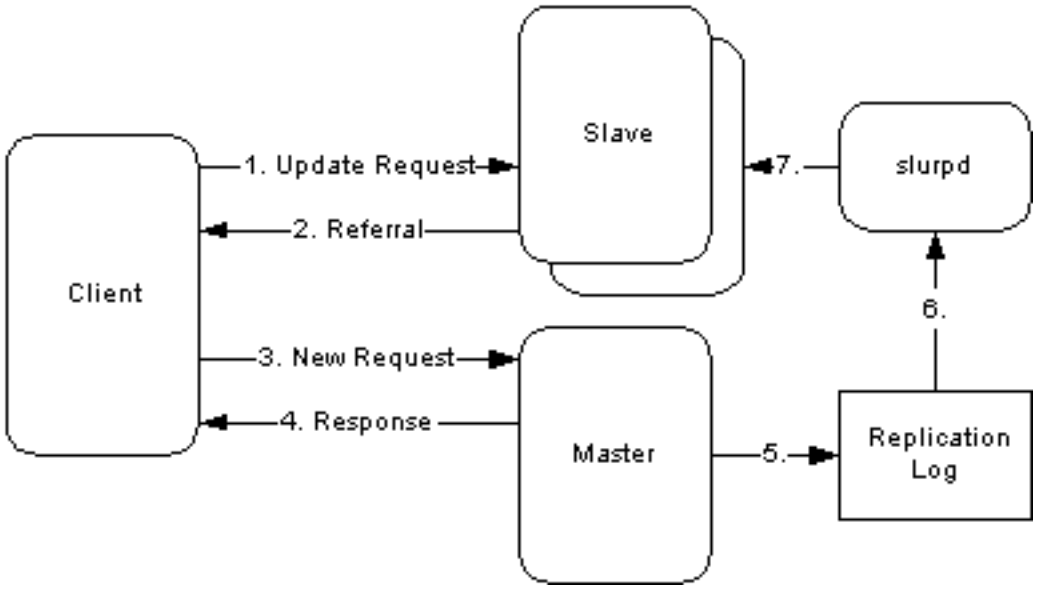

图**3.3:**可复制的目录服务

在单一slapd可靠性不够的情况下，这种配置可以被用来和前两种配置中的一种联合使用。

3.4、 分布式本地目录服务

在这种配置中，本地服务被划分成较小的服务，每一个都可以被复制，并且使用上级和下级引用关联在一起。

4、 编译和安装OpenLDAP软件

本章详细的介绍了如何编译和安装OpenLDAP软件包，包括slapd(8)，独立的LDAP daemon和slurpd(8)——独立的更新复制daemon。编译和安装OpenLDAP需要几个步骤：安装需要的软件，配置OpenLDAP本身，making，以及最终安装。下面的部分详细描述了这个过程。

4.1、 获得并且展开软件

从http://www.openldap.org/software/download/可以下载OpenLDAP。

运行下面的命令：

```
gunzip _c openldap-VERSION.tgz | tar xvfB –

cd openldap-VERSION
```

4.2、 需要的软件

OpenLDAP软件依赖许多第3方的软件包。根据您打算使用的特性，您可能必须下载并且安装许多附加的软件包。本部分详细描述了您必须安装的最经常使用的第3方软件包。注意，某些第3方的软件包也依赖于其他的软件包。按照它们的安装指南安装每一个软件包。

4.2.1、 传输层安全（TLS）

OpenLDAP客户端和服务器需要安装OpenSSL TLS库来提供传输层的安全服务。虽然一些操作系统可能提供了这些库作为基础系统的一部分或者作为一个可选的软件组件，OpenSSL通常需要单独安装。

OpenSSL可以从http://www.openssl.org得到。

除非OpenLDAP的configure检测到了可用的OpenSSL安装，否则，OpenLDAP不是完全符合LDAPV3标准的。

4.2.2、 Kerberos认证服务

OpenLDAP客户端和服务器端支持基于Kerberos的认证服务。特别的，OpenLDAP通过使用

Heimdal或者MIT Kerberos V软件包支持SASL/GSSAPI认证机制。应该安装Heimdal或者MIT Kerberos V。

Heimdal Kerberos 可以从http://www.pdc.kth.se/heimdal/下载，MIT Kerberos可以从http://web.mit.edu/kerberos/www/下载。

强烈推荐使用高强度的认证服务，比如Kerberos提供的认证服务。

4.2.3、 简单认证和安全层

OpenLDAP客户端和服务器需要安装Cyrus ’s SASL库来提供简单认证和安全层服务。虽然一些操作系统可能提供了这些库作为基础系统的一部分或者作为一个可选的软件组件，Cyrus SASL通常需要单独安装。

Cyrus SASL可以从http://asg.web.cmu.edu/sasl/sasl-library.html下载。如果预先安装了OpenSSL和Kerberos/GSSAPI库，Cyrus SASL将使用它们。

除非OpenLDAP的configure检测到了可用的Cyrus SASL安装，否则，OpenLDAP不是完全符合LDAPV3标准的。

4.2.4、 数据库软件

OpenLDAP的slapd(8)主要的后台数据库，BDB，需要Sleepycat Software Berkeley DB，第4版。如果在配置的时候该软件不可用，您将不能编译slapd(8)使用主要的后台数据库。

您的操作系统可能作为基础系统的一部分或者作为一个可选的软件组件提供了Berkeley DB，版本4，如果没有，您需要自己下载并安装。

Berkeley DB可以从http://www.sleepycat.com/download.html下载。有几个可用的版本。现在，推荐使用版本4。

OpenLDAP的slapd(8)LDBM后台数据库支持多种数据库管理器。Berkeley DB 和GDBM。 GDBM 可以从FSF的下载站点ftp://ftp.gnu.org/pub/gnu/gdbm/ 下载。

4.2.5、 线程

OpenLDAP设计上利用了线程。OpenLDAP支持POSIX的pthreads，Mach的Cthreads，和其他一些线程库。如果不能发现合适的线程子系统，configure将会抱怨。如果发生了这种情况，请参考OpenLDAP FAQ http://www.openldap.org/faq/ 的Software|Installation|PlatformHints部分。

4.2.6、 TCP Wrappers

如果事先安装，slapd(8)支持TCP Wrappers（IP层存取控制过滤）。对于一个包含非公共信息的服务期，推荐使用TCP Wrappers或者其他IP层的访问过滤（比如由一个IP层的防火墙提供的）。

4.3、 运行configure

现在，应该使用—help选项运行configure脚本。这将显示一个编译OpenLDAP时您可以改变的可用选项的列表。许多OpenLDAP的特性可以通过这种方法来允许或者禁止。

```
./configure –help
```

configure脚本将检测不同的环境变量。这些环境变量包括：

Table 4.1: Environment Variables

Variable	**Description**
CC	Specify alternative C Compiler
CFLAGS	Specify additional compiler flags
CPPFLAGS	Specify C Preprocessor flags
LDFLAGS	Specify linker flags
LIBS	Specify additional libraries

现在，可以使用需要的配置选项或者环境变量来运行configure。

```
[[env] settings] ./configure [options]
```

例如，假设我们需要使用LDBM后台数据库和支持TCP wrapper来安装OpenLDAP。默认情况下，LDBM是允许的，TCP wrappers是不允许的。因此，我们只需要指明—with-wrappers来包含对TCP Wrapper的支持：

```
./configure --with-wrappers
```

但是，如果所依赖的软件没有安装在系统目录下，上面的命令会失败。比如，如果TCP Wrappers的头文件和库文件安装在/usr/local/include和/usr/local/lib目录，configure脚本应该如下调用：

```
env CPPFLAGS="-I/usr/local/include" LDFLAGS="-L/usr/local/lib" \

./configure --with-wrappers
```

注意：某些**SHELLS**，比如从**Bourne sh(1)**派生的**SHELL**，不需要使用**env(1)**命令。在某些情况下，环境变量必须使用其他的语法指定。

---

configure脚本通常情况下能够自动监测正确的设置。如果您在这个阶段有问题，请参考Platform Specific Hints或者检查您的configure选项。

4.4、 编译软件

当运行了configure脚本之后，最后出现的一行输出应该如下所示：

```
Please "make depend" to build dependencies
```

如果不是，说明configure出现错误，不应该继续。为了编译依赖选项，执行：

```
make depend
```

然后，执行下面的步骤实际编译OpenLDAP：

```
make
```

应该检查上面的命令的输出，已确定所有东西正确编译。注意，这个命令编译了LDAP库，以及LDAP客户端，还有slapd(8)和slurpd(8)。

4.5、 测试软件

当配置和编译成功之后，应该执行测试程序测试编译。执行：

```
make test
```

符合配置的测试程序将运行，并且应该通过。一些测试，比如副职，如果您的配置中不支持，应该跳过。

4.6、 安装软件

成功测试之后就可以准备安装了。必须具有正确的权限。缺省情况下，OpenLDAP安装在/usr/local目录下。如果在configure中使用了—prefix选项改变了目录，将会安装到你指定的目录下。

通常情况下，安装需要超级用户权限。在OpenLDAP的顶层目录，执行：

```
su root -c 'make install'
```

应该仔细检查该命令的输入，确认所有东西正确安装。您将会发现slapd(8)的配置文件默认情况下出现在/usr/local/etc/openldap目录下。请参阅“ SLAPD配置文件 ”一章得到更多的信息。

5、 SLAPD配置文件

成功编译和安装软件之后，可以准备在您的站点中配置slapd(8)了。Slapd运行配置主要是通过slapd.conf(5)文件。通常，该文件安装在/usr/local/etc/openldap目录下。

也可以使用slapd(8)或者slurpd(8)的命令行参数指定其他的配置文件。本章描述了配置文件的格式，并且描述了常用的配置指令。

5.1、 配置文件格式

slapd.conf(5)文件由3个部分的配置信息组成：全局的，特定后台的，和特定数据库的。全局信息首先被指明，后面是特定后台的信息，在后面是特定数据库实例的信息。全局信息可以被特定后台和特定数据库的信息覆盖。特定后台的信息可以被特定数据库的信息覆盖。

空白行和以“ # ”开始的注释行被忽略。如果一行以空格开始，它被认为是上一行的继续。常用的slapd.conf的格式如下所示：

```
# global configuration directives

<global config directives>

# backend definition

backend <typeA>

<backend-specific directives>

# first database definition & config directives

database <typeA>

<database-specific directives>

# second database definition & config directives

database <typeB>

<database-specific directives>
```

```
# second database definition & config directives

database <typeA>

<database-specific directives>

# subsequent backend & database definitions & config directives
```

配置指令可以包含参数。如果包含参数，它们以空格分隔。如果一个参数包含空格，参数应该用双引号因起来，比如“ like this ”。如果一个参数包含双引号或者一个反斜线“ \ ”，它们因该用一个“ \ ”开始。

软件包含了一个示例配置文件，安装在/usr/local/etc/openldap目录下。软件还提供了许多包含模式定义（属性类型attribute types和对象类inject classes）文件，它们在/usr/local/etc/openldap/schema目录下。

5.2、 配置文件指令

本部分详细描述了经常使用的配置指令。要得到完整的列表，请参阅slapd.conf(5)的手册页。本部分将配置文件指令划分为全局的，特定后台的和特定数据的类别，描述了每一个指令和它的默认值，并且给出了使用示例。

5.2.1、 全局指令

本部分描述的指令应用于所有的后台和数据库，除非在后台或者数据库的定义中被明确覆盖。应该被实际文本替换的参数放在尖括号<>中。

5.2.1.1、 access to <what> [by <who> <accesslevel> <control>]+

此指令将访问一组由<what>描述的条目或者属性，并且具有由<accesslevel>指明的存取控制权限，赋予一个或者多个由<who>描述的请求者。参考本章的“ 存取控制 ”部分获得基本的使用信息。

5.2.1.2、 attributetype <RFC2252 Attribute Type Description>

此指令定义了一个属性类型。请参考“ 模式说明 ”一章获取如何使用此指令的更详细的信息

5.2.1.3、 defaultaccess { none | compare | search | read | write }

此指令说明了当没有存取控制指令说明的时候，默认的存取权限。任何给定的存取级别隐含了更

低的级别。比如，读权限隐含了搜索和比较的权限，但是，不包含写权限。

注意：推荐使用**access**指令来设置存取控制。有关**access**指令的详细信息，参阅本章的“ 存取控制 ”部分。

缺省值：

```
defaultaccess read
```

5.2.1.4、 idletimeout <integer>

此指令说明强制关闭一个非活动状态的客户连接之前等待的秒数。默认情况下，该时间等于0，禁止该特性。

5.2.1.5、 include <filename>

此指令说明slapd在继续当前文件的下一行之前，应该从指定的文件读取附加的配置信息。被包含的文件应该符合正常的slapd配置文件的格式。该文件通常被用来包含模式说明。

注意：在使用该指令的时候应该小心。因为对于嵌套的引入指令的最小数量没有限制，并且，没有循环引用的检测。

5.2.1.6、 loglevel <integer>

该指令说明了调试语句和操作统计的日志级别。（当前，日志记录到syslogd(8){EX: LOG_LOCAL4}）应用程序）。在此选项工作之前，必须使用—enable-debug选项配置了OpenLDAP。（除了两个统计级别，它们总是被使用）。日志级别是附加的。要显示那一个数字和什么级别的调试相关联，使用-?运行slapd或者参考下面的表格。可能的<integer>数值包括：

Table 5.1: Debugging Levels

Level	**Description**
-1	enable all debugging
0	no debugging
1	trace function calls
2	debug packet handling
4	heavy trace debugging
8	connection management
16	print out packets sent and received
32	search filter processing

64	configuration file processing
128	access control list processing
256	stats log connections/operations/results
512	stats log entries sent
1024	print communication with shell backends
2048	print entry parsing debugging

示例：

```
loglevel –1
```

将记录许多许多的调试信息。

缺省值：

```
loglevel 256
```

5.2.1.7、 objectclass <RFC2252 Object Class Description>

该指令定义了一个对象类。请参考“ 模式说明 ”一章获得如何使用该指令的更多信息。

5.2.1.8、 referral <URI>

该指令说明了当slapd处理请求时，如果不能在本地数据库中找到信息，将返回的引用。

示例：

```
referral ldap://root.openldap.org
```

这将引用一个非本地的请求到OpenLDAP项目的全局根LDAP服务器。具有智能的LDAP客户端可以向该服务器重新提交请求。但是请注意，大多数客户端只是知道如何处理简单的包含一个主机部分和一个DN（distinguished name）部分的LDAP URL。

5.2.1.9、 sizelimit <integer>

该指令说明了一个搜索操作所能返回的最大条目的数量。

缺省值：

```
sizelimit 500
```

5.2.1.10、 timelimit <integer>

该指令说明了slapd应答搜索请求的时候花费的最长秒数。如果一个请求在该时间内没有结束，会返回一个表示超时的结果。

缺省值：

```
timelimit 3600
```

5.2.2、 通用后端指令

本部分说明的指令只应用于定义它们的后端。它们被所有类型的后端所支持。后端指令应用于所有相同类型的数据库实例，并且，根据不同的指令，可以被数据库指令覆盖。

5.2.2.1、 backend <type>

该指令表示一个后端声明的开始。<type>应该是bdb或者是其他受支持的后端类型中的一种。这些类型在表5.2种列出：

Table 5.2: Database Backends

Types	**Description**
bdb	Berkeley DB transactional backend
dnssrv	DNS SRV backend
ldbm	Lightweight DBM backend
ldap	Lightweight Directory Access Protocol (Proxy) backend
meta	Meta Directory backend
monitor	Monitor backend
passwd	Provides read-only access to *passwd*(5)
perl	Perl Programmable backend
shell	Shell (extern program) backend
sql	SQL Programmable backend
tcp	TCP Programmable backend

示例：

```
database bdb
```

这说明了一个新的BDB后端声明的开始。

5.2.3、 通用数据库指令

本部分说明的指令只应用于定义它们的数据库。它们被每一种类型的数据库所支持。

5.2.3.1、 database <type>

此指令表示一个数据库实例声明的开始。<type>可以是bdb或者是其他受支持的后端类型中的一种。这些类型在表5.2种列出。

示例：

```
database bdb
```

这说明了一个新的数据库实例声明的开始。

5.2.3.2、 readonly { on | off }

此指令将数据库设置为“ 只读 ”状态。任何试图修改数据库的操作将返回“ unwilling to perform ”错误。

缺省值：

```
readonly off
```

5.2.3.3、 replica

```
replica host=<hostname>[:<port>]
        [bindmethod={ simple | kerberos | sasl }]
        ["binddn=<DN>"]
        [mech=<mech>]
        [authcid=<identity>]
        [authzid=<identity>]
        [credentials=<password>]
        [srvtab=<filename>]
```

此指令说明了一个该数据库的一个复制站点。host=参数说明了一个主机和一个可选的端口，此主机上运行了一个附属slapd实例。对于<hostname>可以指定域名或者IP地址。如果没有指定

<port>，使用标准的LDAP端口（389）。

binddn=参数给出了更新附属slapd时绑定的DN。该DN应该具有对附属slapd数据库的读/写权限，通常给定一个在附属slapd的配置文件中指定的rootdn。它必须和附属slapd的配置文件中指出的updatedn相匹配。因为DN很可能会包含空格，因此，整个"binddn=<DN>"字符串应该包含在双引号中。

bindmethod应该是simple或者kerberos或者sasl。取决于当连接到附属slapd时是使用了简单的口令认证，或者是Kerberos认证或SASL认证。

除非存在足够的安全性和秘密保护措施（比如，TLS或者IPSEC），否则，不应该使用简单口令认证。简单口令认证需要声明binddn和credentials参数。

Kerberos认证和SASL认证相比也是不推荐的。尤其是KERBEROS_V4和GSSAPI。Kerberos需要设置binddn和srvtab参数。

SASL认证是最推荐使用的。SASL认证需要使用mech参数指定一种认证机制。根据这种机制，需要分别使用authcid和credentials参数指定一个认证身份和认证证书。authzid参数也可以被用来指定一个认证身份。

请参阅“使用SLURPD进行复制”一章来获得使用此指令的更多信息。

5.2.3.4、 replogfile <filename>

该指令说明了slapd用来记录改变的复制日志文件。复制日志文件通常被slapd写入，并且由slurpd读出。通常情况下，该指令只有在使用slurpd复制数据库的时候才会使用。但是，即使slurpd没有运行，您也可以使用它来产生一个事务记录。在这种情况下，您需要定时截断该文件，否则，它将会持续增长。

请参阅“使用SLURPD进行复制”一章来获得使用此指令的更多信息。

5.2.3.5、 rootdn <dn>

该指令说明了对于该数据库的操作不受限于存取控制或者管理限制的DN。该DN不应该指向目录中的任何条目。该DN可以指向一个SLSL证书。

基于条目的示例：

```
rootdn "cn=Manager,dc=example,dc=com"
```

基于SASL的示例：

```
rootdn "uid=root@EXAMPLE.COM"
```

5.2.3.6、 rootpw <password>

该指令说明了上面的指令中总是能够工作的DN的口令，无论指定的DN条目是否存在，或者，是否具有一个口令。该指令和基于SASL的认证相比是不推荐的。

示例：

```
rootpw secret
```

5.2.3.7、 suffix <dn suffix>

该指令指明了一个传递给后台数据库的查询的DN后缀。可以给出多个后缀行，并且对于每一个数据库定义，至少需要一个。

示例：

```
suffix "dc=example,dc=com"
```

以指定后缀"dc=example,dc=com"结尾的查询将被传递给后台数据库。

注意：当选择了一个传递给后端的查询时，**slapd**在每一个数据库定义中，按照它们在文件中出现的顺序查找后缀行。因此，如果一个数据库后缀是另外一个的前缀，它在配置文件中，必须出现在另外一个的后面。

5.2.3.8、 updatedn <dn>

该指令只有在附属slapd中才会使用。它说明了允许对复本进行改变的DN的名称。这可能是slurpd(8)在对复本做改变的时候绑定的DN，或者是和一个SASL证书相关联的DN。

基于条目的示例：

```
updatedn "cn=Update Daemon,dc=example,dc=com"
```

基于SASL的示例：

```
updatedn "uid=slurpd@EXAMPLE.COM"
```

请参阅“ 使用SLURPD进行复制 ”一章来获得使用此指令的更多信息。

5.2.3.9、 updateref <URL>

该指令只有在附属slapd中才会使用。它说明了当客户端对复本提交更新请求时返回给客户端的URL。如果说明了多次，每一个URL都将被返回。

```
updateref  ldap://master.example.net
```

5.2.4、 BDB数据库指令

该类别的指令只是适用于一个BDB数据库。也就是说，它们必须出现在一个“ database bdb ”行之后，并且在任何后续的“ backend ”或者“ database ”行之前。

5.2.4.1、 directory <directory>

该指令说明了包含BDB数据库文件和相关联的索引的目录。

缺省值：

```
directory /usr/local/var/openldap-data
```

5.2.5、 LDBM数据库指令

该类别的指令只是适用于一个LDBM数据库。也就是说，它们必须出现在一个“ database LDBM ”行之后，并且在任何后续的“ backend ”或者“ database ”行之前。

5.2.5.1、 cachesize <integer>

该指令说明了被LDBM后端数据库实例保持在内存缓存中的条目的数量。

缺省值：

```
cachesize 1000
```

5.2.5.2、 dbcachesize <integer>

该指令说明了和每一个打开的索引文件相关联的内存缓存的字节数。如果它不被底层的数据库方法支持。该指令被忽略。增加该数量将会使用更多的内存，但是，可以使得性能大幅度上升，尤其是在修改或者创建索引的时候。

缺省值：

```
dbcachesize 100000
```

5.2.5.3、 dbnolocking

如果该选项存在，禁止数据库锁定。允许该选项可以提高性能，但是将为数据安全性付出代价。

5.2.5.4、 dbnosync

该选项使得当发生改变时，磁盘上的数据库内容不会立即和内存中的内容进行同步。允许该选项将提高性能，但是，同样将损害数据安全性。

5.2.5.5、 directory <directory>

该指令说明了包含LDBM数据库文件和相关联的索引的目录。

缺省值：

```
directory /usr/local/var/openldap-data
```

5.2.5.6、 index {<attrlist> | default} [pres,eq,approx,sub,none]

该指令说明了对于指定属性维护的索引。如果只是指定一个<attrlist>，将维护缺省的索引。

示例：

```
index default pres,eq

index uid

index cn,sn pres,eq,sub

index objectClass eq
```

第1行设置缺省的索引集合维护为存在和相等，第2行使得对于uid属性类型维护缺省的索引集合（存在和相等）。第3行对于cn和sn属性类型维护存在，子串，和相等索引集合，第4行对于objectClass属性类型维护相等索引。

默认情况下，不维护任何索引。通常情况下，建议对于objectClass属性类型维护一个相等索引。

```
index objectClass eq
```

5.2.5.7、 mode <integer>

该指令说明了对于新创建的数据库索引文件应该具有的保护模式：

缺省值：

```
mode 0600
```

5.3、 存取控制

对于slapd和条目的访问受到存取配置文件指令的控制。一个存取控制行的通常的格式如下：

```
<access directive> ::= access to <what>
   [by <who> <access> <control>]+
<what> ::= * | [ dn[.<target style>]=<regex>]
   [filter=<ldapfilter>] [attrs=<attrlist>]
<target style> ::= regex | base | one | subtree | children
<attrlist> ::= <attr> | <attr> , <attrlist>
<attr> ::= <attrname> | entry | children
<who> ::= [* | anonymous | users | self |
   dn[.<subject style>]=<regex>]
   [dnattr=<attrname> ]
   [group[/<objectclass>[/<attrname>][.<basic style>]]=<regex> ]
   [peername[.<basic style>]=<regex>]
   [sockname[.<basic style>]=<regex>]
   [domain[.<basic style>]=<regex>]
   [sockurl[.<basic style>]=<regex>]
```

[set=<setspec>]

[aci=<attrname>]

<subject style> ::= regex | exact | base | one | subtree | children

<basic style> ::= regex | exact

<access> ::= [self]{<level>|<priv>}

<level> ::= none | auth | compare | search | read | write

<priv> ::= {=|+|-}{w|r|s|c|x}+

<control> ::= [stop | continue | break]

其中，<what>部分选择了该存取控制应用的条目或者属性。<who>部分说明了哪一个身份被赋予访问权限，<access>部分说明了赋予什么样的访问权限。可以指定多个<who> <access> <control>3元组，以允许为相同的条目和属性赋予多个身份不同的访问权限。

5.3.1、 控制对于What的访问

一个存取控制说明的<what>部分决定了存取控制规则应用的条目和属性。条目可以通过两种方式选择：通过匹配条目的DN的正则表达式：

dn=<regular expression>

注意：**DN**的模式说明应该规范到**RFC2253**中限制的**DN**格式。特别是，不应该出现用来分隔不同部分的空格或者逗号。一个规范化的**DN**的例子是：**"cn=Babs Jensen,dc=example,dc=com"**。一个不规范的**DN**的例子是：**"cn=Babs Jensen; dc=example; dc=com"**。

或者，可以通过匹配条目中的某些属性的过滤器来选择条目：

filter=<ldap filter>

<ldap filter>是一个表示一个LDAP搜索过滤的字符串，如RFC2254中描述的那样。

通过在<what>选择器中包含一个用逗号分隔的属性列表来选择条目中的属性：

attrs=<attribute list>

对于条目本身的访问控制必须使用特殊的属性名称“ entry ”来赋予或者拒绝。注意，给一个属性赋予权限是不够的，通过entry属性赋予条目本身访问权限也是需要的。在本部分最后的示例可以帮助您搞清楚这些问题。

最后，有一个特殊的条目选择器“ * ”，该选择器被用来选择任何条目。当没有提供其他的<what>选择器时使用该选择器。它等同于“ dn=.* ”

5.3.2、 赋予Who访问权限

<who>部分指明了被赋予访问权限的身份。注意，权限被赋予“ 身份 ”而不是“ 条目 ”。表5.3说明了身份说明选项：

Table 5.3: Access Entity Specifiers	
Specifier	Entities
*	All, including anonymous and authenticated users
anonymous	Anonymous (non-authenticated) users
users	Authenticated users
self	User associated with target entry
dn=<regex>	Users matching regular expression

DN说明选项使用一个正则表达式来匹配当前身份的“ 规范 ”DN。

```
dn=<regular expression>
```

所谓的“ 规范化 ”，指的是从身份DN中删除多余的空格，并且删除RDN中用来分隔不同部分的逗号。

系统还支持其他的控制方式。比如，一个<what>(By NTKO:此处应该是<who>?)可以通过一个匹配限制客户端的域名：

```
domain=<regular expression>
```

或者通过一个条目，该条目出现在访问权限应用的条目的一个具有一个DN值的属性中：

```
dnattr=<dn-valued attribute name>
```

dnattr指令被用来给予访问一个条目的权限，并且访问者的DN出现该条目的一个具有DN类型的值中。（比如，可以将访问一个组条目的权限给予所有那些列出在该组条目的owner属性中的人。）

5.3.3、 赋予的access权限

可以赋予的<access>的类型可以是如下几种：

Table 5.4: Access Levels		
Level	**Privileges**	**Description**
none		no access
auth	=x	needed to bind
compare	=cx	needed to compare
search	=scx	needed to apply search filters
read	=rscx	needed to read search results
write	=wrscx	needed to modify/rename

每一个级别隐含了更低的级别。因此，给予某人写一个条目的权限同时给了他读，搜索，比较和认证的权限。但是，也可以使用权限说明来赋予特定的权限。

5.3.4、 存取控制计算

当计算某个请求者是否应该被给予访问一个条目或者属性的权限的时候，slapd将条目或者属性和在配置文件中给出的<what>条目或者属性相比较。对于每一个条目，包含该条目的数据库所提供的存取控制被首先应用，（或者如果没有包含在任何一个数据库中，应用第一个数据库的存取控制），然后是全局存取控制指令。在这种优先级的情况下，存取控制按照它们在配置文件中出现的顺序被检查。Slapd在找到第一个匹配该条目或者属性的<what>选择器时停止搜索。相关的存取控制指令就是slapd将要用来计算存取控制的指令。

下一步，slapd将请求访问的身份和上面找到的存取控制指令的<who>相比较，它在找到第一个匹配请求者身份的<who>选择器之后停止。这决定了请求访问者对于条目或者属性所具有的访问权限。

最后，slapd将所选的<access>指令中赋予的权限和客户端请求的权限相比较。如果超过或者等于客户端请求的权限，则赋予该用户权限，否则拒绝访问。

存取控制指令的计算顺序使得它们在配置文件中的顺序非常重要。如果一个存取控制指令在所选择的条目中比另外一个更加特定（选择的范围更小），它应该首先出现在配置文件中。同样，如果一个<who>选择器比另外一个更加特定，它也应该首先出现在存取控制指令中。下面的存取控制示例可以让这个问题更加清楚。

5.3.5、 存取控制示例

上面描述的存取控制应用是非常强大的。下面的部分显示了它们的一些实用示例。首先，几个简单的例子：

```
access to * by * read
```

这个存取控制指令将读权限赋予每一个人。

```
access to *
    by self write
    by anonymous auth
    by * read
```

该指令允许用户修改自己的条目，允许认证，并且允许所有其他人读取。注意，只有第一个by <who>语句匹配应用。因此，匿名用户被赋予认证权限，而非读取权限。因此，最后一个语句和“ by users read ”效果相同。

下面的指令显示了一个使用通过两条存取控制指令中的DN，并且使用正则表达式来选择条目的示例。此示例中，顺序是非常重要的：

```
access to dn=".*,dc=example,dc=com"
     by * search
access to dn=".*,dc=com"
     by * read
```

dc=com子树下面的条目被赋予读权限，除了dc=example,dc=com的子树，它被赋予的是搜索权限。dc=com没有被赋予任何权限，因为没有任何存取控制指令匹配该DN。如果上面的存取控制指令被反过来，后面的存取控制指令将永远不会到达，因为所有dc=example，dc=com的条目同样也是dc=com的条目。

同时注意，如果没有任何access to指令匹配或者没有by <who>语句，访问将被拒绝。也就是说，所有的access to指令都以一个隐含的by * none语句结束，并且每一个存取控制列表以一个隐含的access to * by * none指令结束。只有没有指定存取控制的情况下，才会赋予defaultaccess权限。

下面的例子显示了顺序的重要性，对于存取控制指令和by <who>语句而言都是如此。它也显示了使用一个属性选择器来赋予对特定属性的访问权限，以及不同的<who>选择器。

```
access to dn="(.*,)?dc=example,dc=com" attr=homePhone
     by self write
```

```
    by dn="(.*,)?dc=example,dc=com" search
    by domain=.*\.example\.com read
access to dn="(.*,)?dc=example,dc=com"
    by self write
    by dn=".*,dc=example,dc=com" search
    by anonymous auth
```

该示例适用于所有在“ dc=example,dc=com ”子树中的条目。对于所有除了homePhone以外的属性而言，该条目本身可写，其他example.com条目可以搜索，其他任何人除了认证/授权之外（总是被匿名操作），没有存取权限（由by * none隐含说明）。homePhone属性可以被该条目写，被其他的example.com中的条目搜索，可以被从example.com域中连接的客户读取，其他的将不能读（由by * none隐含说明）。其他的访问将被拒绝（由隐含的access to * by * none说明）。

有时，允许一个特定的DN来从一个属性中增加或者删除自己是很有用的。比如，如果您想创建一个组，并且允许人们想该组的member属性中增加或者删除自己的DN，应该使用如下的存取控制指令来完成这一点：

```
access to attr=member,entry
    by dnattr=member selfwrite
```

这个dnattr说明的<who>选择器说明，该存取控制应用于member属性中列出的条目。selfwrite存取控制选择器说明那些成员只能从属性中增加或者删除自己，而不能增加或者删除其他值。附加的“ entry ”是必须的，因为要访问该条目的属性，能够访问该条目是必须的。

5.4、 配置文件示例

下面是一个配置文件的示例。该示例中有许多说明文字。它定义了两个数据库来处理X.500树中的不同部分；两个部分都是BDB数据库实例。示例中显示的行号只是为了参考，并没有包含在实际的文件中。首先，是全局配置部分：

```
1.  # example config file - global configuration section
2.  include /usr/local/etc/schema/core.schema
3.  referral ldap://root.openldap.org
```

```
4.  access to * by * read
```

第1行是注释。第2行包含了一个包含主要模式定义的配合文件。第3行的referral指令的意思是在后面的本地数据库定义中找不到的查询将被引用到一个运行在标准端口的LDAP服务器。该服务器是root.openldap.org。

第4行是一个全局的存取控制。它应用于所有的条目（在每一个单独的数据库指明的存取控制之后）。

配置文件的下一部分定义了一个BDB后端，它将负责处理“ dc=example,dc=com ”树中的查询。该数据库被复制到两个附属slapd，一个在truelies，另外一个在judgmentday。对于几个属性维护索引，并且，userPassword属性对于非授权访问被保护。

```
5.  # BDB definition for the example.com
6.  database bdb
7.  suffix "dc=example,dc=com"
8.  directory /usr/local/var/openldap-data
9.  rootdn "cn=Manager,dc=example,dc=com"
10.  rootpw secret
11.  # replication directives
12.  replogfile /usr/local/var/openldap/slapd.replog
13.  replica host=slave1.example.com:389
14.        binddn="cn=Replicator,dc=example,dc=com"
15.        bindmethod=simple credentials=secret
16.  replica host=slave2.example.com
17.        binddn="cn=Replicator,dc=example,dc=com"
18.        bindmethod=simple credentials=secret
19.  # indexed attribute definitions
```

```
20.   index uid pres,eq
21.   index cn,sn,uid pres,eq,approx,sub
22.   index objectClass eq
23.   # database access control definitions
24.   access to attr=userPassword
25.         by self write
26.         by anonymous auth
27.         by dn="cn=Admin,dc=example,dc=com" write
28.         by * none
29.   access to *
30.         by self write
31.         by dn="cn=Admin,dc=example,dc=com" write
32.         by * read
```

第5行是注释。第6行的database关键字表示开始数据库定义。第7行制定了传递给该数据库的查询的DN后缀。第8行说明了该数据库文件所处的位置。

第9行和第10行指明了数据库的“超级管理员”，以及他们的口令。该条目不会受限制于存取控制或者大小和时间限制。

第11行到第18行是说明复制的。第12行说明了复制日志文件的位置（该文件记录数据库的改变，被slapd写入，被slurpd读出）。第13行到第15行指明了副本数据库的主机名和端口，和执行改变操作相绑定的DN，以及绑定的方法和执行绑定操作的DN的口令。第16行到第18行说明了第2个复本站点。查看“使用slurpd进行复制”一章来获得有关这些指令的更多信息。

第20行到第22行指明了为哪些属性维护索引。

第24行到第32行说明了对本数据库中条目的存取控制。因为这是第一个数据库，存取控制同样适用于不在任何数据库中保存的条目（比如Root DSE）。对于所有适合的条目，userPassword属性只能被条目本身和“admin”条目写入。它可以被用来进行认证或者授权，但是其他情况时不能读取的。所有其他的条目可以被条目本身和“admin”条目写入，并且可以被其他任何用户读取

（认证过或者没有认证过的用户）。

该示例配置文件的下一个部分定义了另外一个BDB数据库。该数据库查询包含在“ dc=example, dn=net ”子树中的查询。但是，被第一个数据库中相同的条目所管理。注意，如果没有第39行，由于第4行中的全局存取规则，将会允许读取权限。

```
33.  # BDB definition for example.net
34.  database bdb
35.  suffix "dc=example,dc=net"
36.  directory /usr/local/var/openldap-data-net
37.  rootdn "cn=Manager,dc=example,dc=com"
38.  index objectClass eq
39.  access to * by users read
```

6、 运行SLAPD

slapd被设计为可以作为一个单独的服务器运行。这允许服务器充分的利用缓存，利用底层数据库的特性来管理并发，并且节省系统资源。从inetd(8)来运行slapd(8)不能作为选项。

6.1、 命令行选项

slapd(8)支持大量的命令行选项。它们在手册页中有详细的说明。本部分详细说明了几个常用的选项。

-f <filename>

该选项说明了可以为slapd使用的另外的配置文件。缺省选项是正常情况下的/usr/local/etc/openldap/slapd.conf。

-h <URLs>

该选项说明了另外的侦听器配置。缺省的是ldap:///，它表示运行在TCP上的LDAP，在缺省的389端口侦听所有的网络界面。可以指明特殊的主机——端口对或者其他的协议模式（比如，ldaps://或者ldapi://）。例如：-h “ ldaps:// ldap://127.0.0.1:666 ”将创建两个侦听器：一个在默认的LDAP/SSL的636端口，侦听所有网络界面上的通过SSL的LDAP请求，另一个在localhost（环回地址）界面

上侦听666端口。主机可以使用IPV4的点分十进制格式，或者使用主机名称。端口必须是数字。

-n <service-name>

该选项说明了用来进行日志记录或者其他用途的服务名称。默认的服务名称是slapd。

-l <syslog-local-user>

该选项指明了用于syslog(8)应用的本地用户。可以是LOCAL0，LOCAL1，LOCAL2，……LOCAL7等数值。默认值是LOCAL4。该选项并不是在所有系统上都支持。

-u user -g group

该选项分别指明了用来运行的用户或者组的身份。User可以是用户名或者uid，group可以是组名或者gid。

-r directory

该选项指明了一个运行时的目录。slapd将在打开侦听器并且读取任何配置文件或者初始化任何后端数据库之前使用chroot(2)到该目录。

-d <level> | ?

该选项设置slapd的调试级别为<level>。当level是“?”的时候，无论传递给它任何选项，将打印不同的调试级别，然后slapd终止。当前的调试级别是：

Table 6.1: Debugging Levels

Level	**Description**
-1	enable all debugging
0	no debugging
1	trace function calls
2	debug packet handling
4	heavy trace debugging
8	connection management
16	print out packets sent and received
32	search filter processing
64	configuration file processing
128	access control list processing
256	stats log connections/operations/results
512	stats log entries sent
1024	print communication with shell backends
2048	print entry parsing debugging

可以在调试选项中指定多个调试级别。每次指定一个。或者，因为调试级别是可以相加的，可以指定它们的和。比如，如果需要跟踪函数调用，并且查看正在处理的配置文件，可以设置该选项为：-d 65。或者，可以设定为：-d 1 – d 64。参阅<ldap.h>获取更详细的信息。

注意：为了使得除了两个统计级别之外的调试信息生效，**slapd**必须在编译时定义**_DLDAP_DEBUG**。

6.2、 启动SLAPD

通常情况下，slapd可以以如下方式运行：

```
/usr/local/etc/libexec/slapd [<option>]*
```

其中，/usr/local/etc/libexec是由configure脚本决定的，并且<option>是上面描述的选项之一（或者是在slapd(8)中描述）。除非指明了一个调试级别（包括级别0），slapd将会自动fork一个进程，并且从控制它的终端脱离，在后台运行。

6.3、 停止SLAPD

为了安全的停止slapd，应该是用下面的命令：

```
kill -INT `cat /usr/local/var/slapd.pid`
```

其中，/usr/local/var由configure脚本决定。

使用其他的暴力方法停止slapd可能造成信息丢失或者数据库崩溃。

7、 数据库创建和维护工具

本部分告诉您如何从脚本中创建slapd数据库，以及如何在遇到问题的时候定位和解决问题。有两种方法创建数据库。首先，可以联机使用LDAP来创建数据库。通过这种方法，只需要简单的启动slapd，然后，使用LDAP客户端向其中增加条目。该方法对于创建小规模的数据库而言是比较适合的（几百条或者上千条条目）。该方法适合于支持更新的数据库。

第二种创建数据库的方法是使用slapd提供的特殊的应用程序脱机执行。如果您有大量的条目需要创建，使用LDAP的方法将消耗太多的时间，或者如果您想确保数据库在创建的时候没有被访问。注意，不是所有的数据库类型都支持这些应用程序。

7.1、 在LDAP上创建数据库

通过这种方法，您使用LDAP客户端（比如：ldapadd(1)）来增加条目。您应该确认在启动slapd(8)之前在配置文件中设置了如下选项：

```
suffix <dn>
```

如“ 通用数据库指令 ”部分所描述的，该选项定义了该数据库中保存的条目。应该将它设置为将要创建的子树的root DN。例如：

```
suffix "dc=example,dc=com"
```

应该确保指定了索引文件应该被创建的目录：

```
directory <directory>
```

例如：

```
directory /usr/local/var/openldap-data
```

应该使得该目录具有正确的权限，使得slapd可以写入。

还应该配置slapd，以便您可以以具有增加条目权限的用户身份连接。可以配置目录来为该操作指定一个特定的超级管理员用户。这可以通过在数据库定义中使用下面的两个选项实现：

```
rootdn <dn>
```

```
rootpw <passwd>
```

例如：

```
rootdn "cn=Manager,dc=example,dc=com"
```

```
rootpw secret
```

这些选项制定了一个DN和口令，该DN可以被作为超级用户的数据库条目进行认证（该条目允许执行任何操作）。这儿指定的DN和口令总是能够工作，无论该条目的名称实际上是否存在，或者无论其口令是什么。这样就解决了在没有任何条目的情况下如何认证的先有鸡还是先有蛋的问题。

最后，还应该确保数据库定义包含了需要的索引定义：

```
index {<attrlist> | default} [pres,eq,approx,sub,none]
```

例如，要索引cn,sn,uid和objectclass属性，可以使用下面的索引指令：

```
index cn,sn,uid

index objectClass pres,eq
```

注意，并不是所有的索引类型对于多有的属性类型都是可用的。参阅“SLAPD配置文件”部分查看该选项的更详细的信息。配置完毕之后，启动slapd，使用LDAP客户软件连接到LDAP，开始增加条目。比如，要使用ldapadd工具增加一个组织条目和一个组织角色条目，应该创建一个包含如下所示内容的entries.ldif文件：

```
# Organization for Example Corporation

dn: dc=example,dc=com

objectClass: dcObject

objectClass: organization

dc: example

o: Example Corporation

description: The Example Corporation

# Organizational Role for Directory Manager

dn: cn=Manager,dc=example,dc=com

objectClass: organizationalRole

cn: Manager

description: Directory Manager
```

然后，使用下面的命令创建条目：

```
ldapadd -f entries.ldif -x -D "cn=Manager,dc=example,dc=com" -w secret
```

上面的命令假设按照上面例子中的配置。

7.2、 脱机创建数据库

创建数据库的第二种方法是使用下面描述的slapd数据库工具脱机创建。如果有大量的条目需要创建，该方法是最好的。这些工具读取slapd的配置文件，和一个包含要增加的条目的文本描述的输入文件。对于支持这些工具的数据库类型而言，它们直接创建出数据库文件。（否则，您必须使用上面提到的方法）。首先，您必须在数据库定义的配置文件中设置几个非常重要的配置选项：

```
suffix <dn>
```

正如在“ 通用数据库指令 ”部分所描述的，该选项设置了该数据库保存哪些条目。你应该将该选项设置为您要创建的子树的root DN。例如：

```
suffix "dc=example,dc=com"
```

应该确保指定了索引文件应该被创建的目录：

```
directory <directory>
```

例如：

```
directory /usr/local/var/openldap-data
```

最后，还应该确保数据库定义包含了需要的索引定义。这可以通过一个或者几个index指令来实现：

```
index {<attrlist> | default} [pres,eq,approx,sub,none]
```

例如，要索引cn,sn,uid和objectclass属性，可以使用下面的索引指令：

```
index cn,sn,uid

index objectClass pres,eq
```

这为cn，sn，和uid创建了存在，相等和子串索引，为objectclass属性创建了相等索引。参与配置文件部分获得更多信息。

7.2.1、 slapadd程序

在配置好所有设置之后，您可以运行slapadd程序来创建主要的数据库及其索引。

```
slapadd -l <inputfile> -f <slapdconfigfile>
```

[-d <debuglevel>] [-n <integer>|-b <suffix>]

其参数的含义如下：

-l <inputfile>

说明了包含要增加的条目的文本格式的LDIF输入文件（参阅后面的“ LDIF文本条目格式 ”部分）。

-f <slapdconfigfile>

说明了slapd配置文件的格式。该配置文件说明了在何处创建索引，以及创建什么索引等等。

-d <debuglevel>

打开调试。如<debuglebel>所说明的。调试级别和slapd相同。参阅“ 运行SLAPD ”中的“ 命令行选项 ”部分。

-n <databasenumber>

一个说明修改那一个数据库的可选参数。第1个列在配置文件中的数据库是1，第2个是2……等等。默认情况下，使用配置文件中的第1个数据库。该选项不能和-b参数一起使用。

-b <suffix>

一个说明修改那一个数据库的可选参数。指定的后缀和数据库suffix指令相匹配来获得数据库编号。不应该和-n一起使用。

7.2.2、 slapindex程序

有时重新安排索引是必须的（比如修改了slapd.conf(5)之后）。可以通过使用slapindex(8)程序来实现。slapindex运行方法如下：

slapindex -f <slapdconfigfile>

[-d <debuglevel>] [-n <databasenumber>|-b <suffix>]

其中，-f –d –n和-b选项和slapadd程序中的含义相同。Slapindex根据当前数据库的内容重新创建索引。

7.2.3、 slapcat程序

该程序将数据库转储到一个LDIF文件。当您创建一个您的数据库的可被人工读取的拷贝时，或者当您想脱机编辑您的数据库时，该程序是非常有用的。该程序这样运行：

```
slapcat -l <filename> -f <slapdconfigfile>

    [-d <debuglevel>] [-n <databasenumber>|-b <suffix>]
```

其中-n和-b参数用来选择由-f参数指定的slapd.conf(5)文件中的配置数据库。相关的LDIF文件输出到标准输出或者输出到使用-l选项指明的文件。

7.3、 LDIF文本条目格式

LDAP数据交换格式，LDIF，被用来使用简单的格式来表示LDAP条目。本部分提供了一个LDIF条目格式的一个简要描述。可以作为ldif(5)和RFC2849的补充。

一个条目的基本格式是：

```
# comment

dn: <distinguished name>

<attrdesc>: <attrvalue>

<attrdesc>: <attrvalue>

...
```

以#开头的行表示注释。一个属性描述可以是一个简单的属性类型，比如cn或者objectClass或者1.2.3（和一个属性类型相关联的OID）或者可以包含选项。比如。Cn;lang_en_us或者userCertificate;binary。

一行的积蓄可以以一个空格或者一个TAB键开始下一行。例如：

```
dn: cn=Barbara J Jensen,dc=example,dc=

 com

cn: Barbara J

 Jensen
```

和：

```
dn: cn=Barbara J Jensen,dc=example,dc=com

cn: Barbara J Jensen
```

是相同的。

多个属性值可以在不同的行中指定。比如：

```
cn: Barbara J Jensen

cn: Babs Jensen
```

如果一个<attrvalue>包含不可打印的字符，或者以一个空格，一个冒号（“ ： ”），或者一个小于号（“ < ”）开始，<attrdesc>后面跟着两个冒号，和其值的BASE64编码。例如，值“ begins with a space ”应该被如下编码：

```
cn:: IGJlZ2lucyB3aXRoIGEgc3BhY2U=
```

也可以指定一个包含属性值的URL。比如，下面说明了jpegPhoto属性值应该从文件file:///path/to/file.jpeg得到：

```
cn:< file:///path/to/file.jpeg
```

一个LDIF文件中的多个条目以空行分隔。下面是一个包含3个条目的LDIF文件的例子：

```
# Barbara's Entry

dn: cn=Barbara J Jensen,dc=example,dc=com

cn: Barbara J Jensen

cn: Babs Jensen

objectClass: person

sn: Jensen

# Bjorn's Entry

dn: cn=Bjorn J Jensen,dc=example,dc=com
```

```
cn: Bjorn J Jensen

cn: Bjorn Jensen

objectClass: person

sn: Jensen

# Base64 encoded JPEG photo

jpegPhoto:: /9j/4AAQSkZJRgABAAAAAQABAAD/2wBDABALD

 A4MChAODQ4SERATGCgaGBYWGDEjJR0oOjM9PDkzODdASFxOQ

 ERXRTc4UG1RV19iZ2hnPk1xeXBkeFxlZ2P/2wBDARESEhgVG

# Jennifer's Entry

dn: cn=Jennifer J Jensen,dc=example,dc=com

cn: Jennifer J Jensen

cn: Jennifer Jensen

objectClass: person

sn: Jensen

# JPEG photo from file

jpegPhoto:< file:///path/to/file.jpeg
```

注意，Bjorn条目的jpegPhoto属性是BASE64编码的，Jeniffer条目的jpegPhoto属性则是从URL指定的文件中获取的。

注意：多余的空格在**LDIF**文件中没有从值中删除。多个中间的空格也没有被压缩。如果你不想让它们出现在数值中，不要将它们留在哪儿。

8、 模式说明

本章描述了如何扩展slapd(8)使用的用户模式。第1个部分，“ 已发布的模式文件 ”，详细说明了

软件发布版本中提供的可选的模式定义，以及从何处获取其他的定义。第2个部分，“扩展模式”，说明了如何定义新的模式项目。

8.1、已发布的模式文件

OpenLDAP岁软件发布了一组模式定义，您可以直接使用它们。每一组模式定义为可以被包含到您的slapd.conf(5)中的文件。（可以使用include指令）这些模式文件在正常情况下安装在/usr/local/etc/openldap/schema目录下。

Table 8.1: Provided Schema Specifications

File	Description
core.schema	OpenLDAP *core* (required)
cosine.schema	Cosine and Internet X.500 (useful)
inetorgperson.schema	InetOrgPerson (useful)
misc.schema	Assorted (experimental)
nis.schema	Network Information Services (FYI)
openldap.schema	OpenLDAP Project (experimental)

要使用其中的任何模式文件，只需要在您的slapd.conf(5)文件中的全局定义部分包含需要的文件。比如：

```
# include schema

include /usr/local/etc/openldap/schema/core.schema

include /usr/local/etc/openldap/schema/cosine.schema

include /usr/local/etc/openldap/schema/inetorgperson.schema
```

其他的文件也可以获得。请参考OpenLDAP FAQ（http://www.openldap.org/faq/）。

8.2、扩展模式

slapd(8)中使用的模式可以被扩展为支持其他的语法，匹配规则，属性类型和对象类。本章详细说明了如何使用slapd已经支持的语法和匹配规则来增加用户应用属性类型和对象属性类型。slapd也可以被扩展为支持其他的语法，匹配规则和系统模式，但是，这需要某种程度的编程，因此，在此处不予讨论。

下面是定义一个新的模式的5个步骤：

- 获得对象标识；
- 选择一个名称前缀；
- 创建本地模式文件；
- 定义自定义属性类型；（如果必须）
- 定义自定义对象类。

8.2.1、 对象标识

每一个模式元素由一个全局唯一对象标识符（OID）标识。OID同时被用来标识其他的对象。他们通常可以在ASN.1描述的协议中找到。特别是，它们在简单网络管理协议（SNMP）中被广泛使用。因为OID是层次的，您的组织可以获得一个OID，并且在需要的时候对其进行分支扩展。比如，如果您的组织被赋予一个OID1.1，您可以按照如下所示的方法扩展树：

Table 8.2: Example OID hierarchy

OID	**Assignment**
1.1	Organization's OID
1.1.1	SNMP Elements
1.1.2	LDAP Elements
1.1.2.1	AttributeTypes
1.1.2.1.1	myAttribute
1.1.2.2	ObjectClasses
1.1.2.2.1	myObjectClass

您当然可以在您的组织的OID下面自由的根据组织的需要设计您的层次结构。无论您选择了怎样的层次，您应该保持一份分配注册表。这可以是一个简单的平面文件或者是一个更加复杂的东西，比如，OpenLDAP OID Registry（http://www.openldap.org/faq/index.cgi?file=197）。

有关对象标识符（和一个服务列表）的更多信息，请参阅：

http://www.alvestrand.no/harald/objectid/。

任何情况下，您都不应该使用一个伪造的OID!

为了免费得到一个注册过的OID，在Internet Assigned Numbers Authority维护的Private Enterprise arch下申请一个OID。任何私人企业或者组织可以申请一个在此arch下的OID。只需要填写一个位于http://www.iana.org/cgi-bin/enterprise.pl的IANA表单就可以了。您的合法OID将在几天内发送给您。您的基ID将是类似于1.3.6.1.4.1.X，其中X是一个整数。

注意：不要让**IANA**页面上的“ **MIB/SNMP** ”声明混淆您的视线。从这个表单中申请的**OID**可以用于任何用途，包括标识**LDAP**模式元素。

8.2.2、 名称前缀

除了给每一个模式元素分配一个唯一的对象标识符，您应该给每一个模式元素提供一个文本名称。该名称应该是既有描述性，又不会和其他的模式名称冲突。特别的，任何您选择的名称都不应该和已经有的或者是将要使用的标准名称冲突。

为了减少（但不是消除）潜在的名称冲突，一个简便的方法是在非标准名称前增加几个字母的前缀，来将组织的改变本地化。组织名称越短，就应该提供越长的前缀。

在下面的示例中，我们选择了一个很短的前缀“ my ”来减少空间。在一个大型的，全球性的组织中，使用这样短的前缀是不合适的。对于一个小的，本地的公司，我们推荐象“ deFirm ”（德国公司）或者“ comExample ”（和example.com关联的组织的元素）。

8.2.3、 本地模式文件

配置文件指令中的objectClass和attibuteTypes可以被用来定义目录中的模式规则。习惯上创建一个文件来包含对于定制的模式元素的定义。我们推荐您在目录/usr/local/etc/openldap/schema/local.schema中创建一个local.schema文件，然后，在您的slapd.conf(5)文件中的其他模式包含指令之后包含该文件。

```
# include schema

include /usr/local/etc/openldap/schema/core.schema

include /usr/local/etc/openldap/schema/cosine.schema

include /usr/local/etc/openldap/schema/inetorgperson.schema

# include local schema

include /usr/local/etc/openldap/schema/local.schema
```

8.2.4、 属性类型说明

指令attributetype被用来定义一个新的属性类型。该指令和在子模式子树中的attributeTypes属性使用相同的属性类型描述（如RFC2252所定义）。比如：

attributetype <RFC2252 Attribute Type Description>

其中，Attribute Type Description按照如下BNF定义：

```
AttributeTypeDescription = "(" whsp
    numericoid whsp           ; AttributeType identifier
  [ "NAME" qdescrs ]          ; name used in AttributeType
  [ "DESC" qdstring ]         ; description
  [ "OBSOLETE" whsp ]
  [ "SUP" woid ]              ; derived from this other
                        ; AttributeType
  [ "EQUALITY" woid           ; Matching Rule name
  [ "ORDERING" woid           ; Matching Rule name
  [ "SUBSTR" woid ]           ; Matching Rule name
  [ "SYNTAX" whsp noidlen whsp ] ; Syntax OID
  [ "SINGLE-VALUE" whsp ]     ; default multi-valued
  [ "COLLECTIVE" whsp ]       ; default not collective
  [ "NO-USER-MODIFICATION" whsp ]; default user modifiable
  [ "USAGE" whsp AttributeUsage ]; default userApplications
  whsp ")"
AttributeUsage =
  "userApplications"    /
  "directoryOperation"  /
  "distributedOperation" / ; DSA-shared
```

```
"dSAOperation"      ; DSA-specific, value depends on server
```

其中，whsp是一个空格，numericoid是一个全局唯一的点分十进制格式的OID（例如：1.1.0），qdescrs是一个或者多个名称，woid或者是OID的名称，或者是OID后面加上可选的长度说明（比如：{10}）。

例如，属性类型name和cn在core.schema中如下定义：

```
attributeType ( 2.5.4.41 NAME 'name'
    DESC 'name(s) associated with the object'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
attributeType ( 2.5.4.3 NAME ( 'cn' $ 'commonName' )
    DESC 'common name(s) asscіated with the object'
    SUP name )
```

请注意，每一个都定义了属性的OID，给出了一个短的名称，以及一个简短的描述。每一个名称都是OID的一个别名。Slapd(8)在返回结果的时候，将返回第1个列出的名称。

第1个名称，name，保存了directoryString（UTF-8编码的Unicode）语法。该语法由OID说明。（1.3.6.1.4.1.1466.115.121.1.15标识了目录字符串语法）。还说明了一个推荐长度为32768的选项。服务器应该支持该长度的值，但是，也可以支持更长的值。该域没有指明长度限制，因此，在服务器上被忽略，并且服务器不会限制其大小。另外，相等和子串匹配使用不区分大小写的规则。下面是经常使用的语法和匹配规则（OpenLDAP支持这些，以及更多）

Table 8.3: Commonly Used Syntaxes

Name	OID	Description
boolean	1.3.6.1.4.1.1466.115.121.1.7	boolean value
distinguishedName	1.3.6.1.4.1.1466.115.121.1.12	DN
directoryString	1.3.6.1.4.1.1466.115.121.1.15	UTF-8 string
IA5String	1.3.6.1.4.1.1466.115.121.1.26	ASCII string
Integer	1.3.6.1.4.1.1466.115.121.1.27	integer
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	DN plus UID

Numeric String	1.3.6.1.4.1.1466.115.121.1.36	numeric string
OID	1.3.6.1.4.1.1466.115.121.1.38	object identifier
Octet String	1.3.6.1.4.1.1466.115.121.1.40	arbitary octets
Printable String	1.3.6.1.4.1.1466.115.121.1.44	printable string

Table 8.4: Commonly Used Matching Rules

Name	**Type**	**Description**
booleanMatch	equality	boolean
octetStringMatch	equality	octet string
objectIdentiferMatch	equality	OID
distinguishedNameMatch	equality	DN
uniqueMemberMatch	equality	Name with optional UID
numericStringMatch	equality	numerical
numericStringOrderingMatch	ordering	numerical
numericStringSubstringsMatch	substrings	numerical
caseIgnoreMatch	equality	case insensitive, space insensitive
caseIgnoreOrderingMatch	ordering	case insensitive, space insensitive
caseIgnoreSubstringsMatch	substrings	case insensitive, space insensitive
caseExactMatch	equality	case sensitive, space insensitive
caseExactOrderingMatch	ordering	case sensitive, space insensitive
caseExactSubstringsMatch	substrings	case sensitive, space insensitive
caseIgnoreIA5Match	equality	case insensitive, space insensitive
caseIgnoreIA5OrderingMatch	ordering	case insensitive, space insensitive
caseIgnoreIA5SubstringsMatch	substrings	case insensitive, space insensitive
caseExactIA5Match	equality	case sensitive, space insensitive
caseExactIA5OrderingMatch	ordering	case sensitive, space insensitive
caseExactIA5SubstringsMatch	substrings	case sensitive, space insensitive

第2个属性，cn，是name的一个子类型，因此，它继承了语法，匹配规则，并且使用name. commonName作为别名。

两个属性都没有限制到单一值。都可以被用户应用程序所使用，都不存在过期，都不是集合。

下面的部分给出了几个示例。

8.2.4.1、 myUniqueName

许多组织为每个用户维护了一个单一的唯一名称。虽然可以使用displayName（RFC2798），该属

性实际上意味着让用户控制，而不是被组织控制。我们可以简单的从inetorgperson.schema中拷贝displayName的定义，然后替换掉OID，name，以及description。比如：

```
attributetype ( 1.1.2.1.1 NAME 'myUniqueName'
        DESC 'unique name with my organization'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )
```

但是，如果我们需要该名字被包含在name声明中[比如，(name=*Jane*)]，该属性可以被定义为name的一个子类型。比如：

```
attributetype ( 1.1.2.1.1 NAME 'myUniqueName'
        DESC 'unique name with my organization'
        SUP name )
```

8.2.4.2、 myPhoto

许多组织包含了每一个用户的照片。可以定义一个myPhoto属性类型来保存照片。当然，可以使用jpegPhoto(RFC2798)或者其子类型来保存照片。但是，只有在照片是JPEG格式的情况下才能这样做。作为一种替代方法，可以定义一个使用Octet String语法的属性类型。比如：

```
attributetype ( 1.1.2.1.2 NAME 'myPhoto'
        DESC 'a photo (application defined format)'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
        SINGLE-VALUE )
```

这样，该语法没有指明照片的格式。它假设（可能不正确）所有访问该属性的应用统一处理该值。

如果需要支持多种图片格式，应该为每一种格式定义一个另外的属性类型，为照片提供类型信

息，或者使用ASN.1描述值，并且使用;binary传输选项。

另外一个选择是让属性保存一个指向图片的URI。可以在labeledURI(RFC2079)之后定义该属性，或者只是简单的创建一个子类型。比如：

```
attributetype ( 1.1.2.1.3 NAME 'myPhotoURI'
        DESC 'URI and optional label referring to a photo'
        SUP labeledURI )
```

8.2.5、 对象类说明

objectclasses指令用来定义一个新的对象类。该指令和在子模式子树中的objectClasses属性使用相同的属性类型描述（如RFC2252所定义）。比如：

```
objectclass <RFC2252 Object Class Description>
```

其中，Object Class Description按照如下所示的BNF定义：

```
ObjectClassDescription = "(" whsp
        numericoid whsp      ; ObjectClass identifier
        [ "NAME" qdescrs ]
        [ "DESC" qdstring ]
        [ "OBSOLETE" whsp ]
        [ "SUP" oids ]       ; Superior ObjectClasses
        [ ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) whsp ]
                ; default structural
        [ "MUST" oids ]      ; AttributeTypes
        [ "MAY" oids ]       ; AttributeTypes
        whsp ")"
```

其中，whsp是一个空格，numericoid是一个全局唯一的点分十进制格式的OID（例如：1.1.0），qdescrs是一个或者多个名称，oids是一个或者多个OID的名称。

8.2.5.1、 myPhotoObject

下面定义一个允许将myPhoto增加到任何已经存在的条目中的auxiliary对象类：

```
objectclass ( 1.1.2.2.1 NAME 'myPhotoObject'
        DESC 'mixin myPhoto'
        AUXILIARY
        MAY myPhoto )
```

8.2.5.2、 myPerson

如果您的组织需要一个私有的结构化对象类来表示用户，你可以子类化任何一个已经存在的person类，比如inetOrgPerson(RFC2798)，然后增加需要的属性：

```
objectclass ( 1.1.2.2.2 NAME 'myPerson'
        DESC 'my person'
        SUP inetOrgPerson
        MUST ( 'myUniqueName' $ 'givenName' )
        MAY 'myPhoto' )
```

该对象类从inetOrgPerson中继承允许的或者必须的属性，但是，要求myUniqueName和givenName，允许myPhoto。

8.2.6、 OID宏

为了简化OID的管理和使用，slapd(8)支持对象描述宏。objectIdentifier用来将一个宏名连接到一个OID。OID可以是从一个先前已经定义的OID宏中派生的。slapd.conf(5)的语法是：

```
objectIdentifier <name> { <oid> | <name>[:<suffix>] }
```

下面显示了定义一组OID宏，并且在定义模式元素中使用它们：

```
objectIdentifier myOID  1.1

objectIdentifier mySNMP myOrgOID:1

objectIdentifier myLDAP myOrgOID:2

objectIdentifier myAttributeType       myOrgLDAP:1

objectIdentifier myObjectClass  myOrgLDAP:2

attributetype ( myAttributeType:3 NAME 'myPhotoURI'

      DESC 'URI and optional label referring to a photo'

      SUP labeledURI )

objectclass ( myObjectClass:1 NAME 'myPhotoObject'

      DESC 'mixin myPhoto'

      AUXILIARY

      MAY myPhoto )
```

9、 使用SASL

OpenLDAP客户端和服务器能够通过简单认证和SASL框架(在RFC2222中详细描述)来认证用户。本章描述了如何在OpenLDAP中使用SASL。

SASL可以和几种工业标准的认证机制一起工作。包括Kerberos V4，GSSAPI，已经其他的一些摘要机制。标准的OpenLDAP提供的客户端工具，比如ldapsearch(1)以及ldapmodify(1)，默认情况下将试图使用SASL来向slapd(8)服务器认证用户。基本的认证服务可以让LDAP管理员在几个步骤中设置好，允许用户以他们的LDAP条目的身份向slapd服务器进行认证。通过另外几个步骤，用户和服务可以利用SASL的授权机制，允许他们认证自己，然后，切换他们的身份到其他的用户或者服务。

本章假设您已经阅读过了Cyrus SASL系统管理指南，该指南和Cyrus SASL软件包一起提供（在/doc/sysadmin.html）。

注意，下面的文本中，术语“用户”来描述一个通过一个LDAP客户端（比如ldapsearch(1)）连接到LDAP服务器的人或者应用程序实体。也就是说，术语“用户”不只是使用LDAP客户端的个人，还有不同过用户的直接控制而发起LDAP客户端操作的应用程序实体。比如，一个使用LDAP

来获取存放在一个LDAP服务器中的信息的电子邮件服务器就是一个应用程序实体。

9.1、 SASL安全考虑

SASL提供了多种不同的认证机制。本部分简要列出安全方面的考虑。

一些人正机制，比如PLAIN和LOGIN，没有提供比LDAP的“ 简单 ”认证更高的安全性。和“ 简单 ”认证一样，除非您有足够的安全保护措施，不应该使用这种机制。推荐这种机制只是和传输层安全（TLS）一起使用。本文档将不再讨论使用PLAIN和LOGIN机制。

DIGEST-MD5机制是LDAPV3中强制要求实现的认证机制。虽然DIGEST-MD5和可信第3方的认证系统（比如Kerberos或者公用密钥系统）比较起来，没有提供足够强度的安全，但是，它的确提供了一个能够防护多种攻击的保护措施。和CRAM-MD5 的机制不同，它防止了选择明文攻击。DIGEST-MD5比脆弱的甚至是危险的使用明文口令的机制要好得多。和DIGEST-MD5比较，CRAM-MD5要差一些。下面会讨论使用DIGEST-MD5。

KERBEROS_V4认证机制使用了Kerberos IV来提供安全的认证服务。还有基于GSSAPI的认证机制，主要用来和Kerberos V一起使用。Kerberos被认为是一个安全的，分布式的认证系统，并且适合于小规模和大规模的企业。下面会讨论使用KERBEROS_V4和GSSAPI。

EXTERNAL机制利用了底层网络服务（比如TLS）提供的认证服务。当和TLS 基于X.509的公用密钥系统一起使用时，EXTERNAL机制提供了高强度的认证机制。在“ 使用TLS ”一章中将讨论使用EXTERNAL。

还有其他的强度认证机制可以选择，包括OTP（one time passwords）和SRP（secure remote passwords）。这些机制在本文档中没有讨论。

9.2、 SASL认证

运行基本的SASL认证需要几个步骤。第1步是配置slapd服务器环境，让它可以和客户端的程序使用您站点的安全系统进行通信。这通常包括设置一个服务密钥，一个公用密钥，或者其他形式的秘密。第2步的重点在于将认证身份映射到LDAP的DN，这取决于目录中的条目是怎样组织的。下一部分将给出一个使用Kerberos V4作为示例机制来完成第1个步骤的解释。将您的站点配置为使用验证机制的步骤是很相似的。但是，本章没有提供SASL中包含的每一种认证机制的指南。再下一个部分则描述了将认证身份映射到DN的步骤。

9.2.1、 GSSAPI

本部分描述了使用和OpenLDAP一起使用SASL的GSSAPI机制和Kerberos V。假设已经部署了Kerberos V，并且您熟悉系统的操作，而且，您的用户已经被培训过使用系统。本部分还假设您已经通过阅读*Configuring GSSAPI和Cyrus SASL* (和Cyrus SASL一起提供，位于doc/gssapi文件)，并

且熟悉了使用GSSAPI的机制，并且，成功的试验过Cyrus提供的样例服务器和样例客户端程序。有关Kerberos的更多的信息，可以从http://web.mit.edu/kerberos/www/获取。

为了和slapd(8)一起使用GSSAPI，必须首先创建一个服务密钥，该密钥应该具有一个服务所运行的域中的主机域的ldap主题词。比图，如果您在directory.example.com上运行slapd，并且您的域是EXAMPLE.COM，您需要创建一个服务密钥，该密钥使用下面的主题词：

```
ldap/directory.example.com@EXAMPLE.COM
```

当slapd(8)运行的时候，它必须有权访问该密钥。通常情况下，可以将该key放置到一个keytab中。比如：/etc/krb5.keytab。

为了使用GSSAPI认证机制来认证目录，用户在运行LDAP客户端之前首先获得一个Ticket Granting Ticket（TGT）。当使用OpenLDAP客户端工具的时候，用户可以通过在命令行中指定-Y GSSAPI选项来要求使用GSSAPI认证机制。

为了认证和授权，slapd(8)和一个没有映射的认证DN相关联，该DN的格式如下：

```
uid=principal,cn=GSSAPI,cn=auth
```

如果用户的主题词在同一个域中，该域从主题词中删除。继续我们的示例，一个具有Kerberos主题词kurt@EXAMPLE.COM的用户将具有下面的DN：

```
uid=kurt,cn=GSSAPI,cn=auth
```

主题词ursula@@FORIEGN.REALM将具有下面的DN：

```
ursula@@FORIEGN.REALM
```

9.2.2、 KERBEROS_V4

该部分描述了和OpenLDAP一起使用SASL KERBEROS_V4认证机制。假设您已经熟悉了Kerberos IV安全系统的工作过程，并且您的站点已经部署了Kerberos IV。您的用户应该熟悉认证策略，并且知道如何从一个Kerberos ticket cache中接受证书，以及如何更新将要到期的证书。

客户端程序在连接到您得LDAP服务器的时候，需要能够获取一个会话密钥，这允许LDAP服务器知道用户的身份，并且允许客户端知道，它正在连接一个正确的服务器。如果使用了加密层，会话密钥同样可以用来帮助实现握手。

Slapd服务器运行叫做“ ldap ”的服务，并且服务器需要一个带有服务密钥的srvtab文件。支持SASL的客户端程序将获取一个带有该用户的ticket granting ticket(TGT)的“ ldap ”服务ticket，并且

该ticket匹配OpenLDAP服务器的主机名。例如，如果您的域叫做EXAMPLE.COM，并且slapd服务器运行在主机directory.example.com上，服务器上的/etc/srvtab文件将具有一个服务密钥：

```
ldap.directory@EXAMPLE.COM
```

当一个LDAP客户端使用KERBEROS_IV向目录进行验证的时候，它将请求一个相同主题的会话密钥。要么从ticket cache，要么从Kerberos服务器获取一个新的。这就要求该TGT在cache中是可用的并且是正确的。如果不存在或者已经过期，SASL将打印出下面的信息：

```
ldap_sasl_interactive_bind_s: Local error
```

当获取了服务ticket之后，它将被传递给LDAP服务器用来证明用户的身份。服务器将使用SASL库调用，从服务ticket中解析出身份和域信息，并将它们转换成一个如下格式的验证请求DN：

```
uid=<username>,cn=<realm>,cn=<mechanism>,cn=auth
```

因此，在我们上面的例子中，如果用户名称是“ adamson ”，认证请求DN将是：

```
uid=ADAMSON,cn=EXAMPLE.COM,cn=KERBEROS_V4,cn=AUTH
```

这个认证请求DN可以被放到ACL中的或者groupOfNames的“ member ”属性中。因为它是合法的LDAP DN格式。下一部分将阐述如何将该DN映射到个人自己的LDAP条目的DN。

同时注意在该示例中，因为是Kerberos，显示了DN中的<realm>部分被使用公司的Kerberos域进行填充。几个其他的认证机制不使用域的概念，因此，认证请求DN中的“ ,cn=<realm> ”部分不会出现。

9.2.3、 将认证身份映射到LDAP条目

slapd服务器中的认证机制将基于底层使用的认证机制，使用SASL库调用来获得已认证用户的用户名。该用户名是在认证机制的名字空间中。不是在LDAP空间中。正如上面的部分所显示的，用户名被重新格式化为如下格式的一个认证请求DN：

```
uid=<username>,cn=<realm>,cn=<mechanism>,cn=auth
```

或者：

```
uid=<username>,cn=<mechanism>,cn=auth
```

取决于认证机制<mechanism>是否使用了“ 域 ”的概念。

这并不意味着您应该将上述LDAP条目增加到您的LDAP数据库中。如果您对于每一个将要在

LDAP中验证的人都在LDAP目录树中有一个LDAP条目，并且该树不是以cn=auth开始。但是，如果您的站点用户名和LDAP中该用户的条目之间有一个清晰的映射，您可以配置您的LDAP服务器自动映射一个用户的验证用户名到他们的验证DN。

LDAP管理员将需要告诉LDAP服务器，如何映射一个认证请求DN到一个用户的认证DN。这可以通过在slapd.conf(5)文件中增加一个或者多个saslRegexp指令来实现。该指令使用两个参数：

```
saslRegexp  <search pattern>  <replacement pattern>
```

认证请求DN和使用正则表达式函数regcomp()和regexec()，与<search pattern>相比较。如果匹配，它将被使用<replacement pattern>重写。如果有多个saslRegexp指令，只使用第一个匹配<search pattern>的认证身份。从<replacement pattern>输出的字符串应该是用户的认证DN，使用合法的LDAP DN格式。它也可以是一个LDAP URL，在下面的部分讨论。

search pattern可以包含在regexec(3C)中列出的任何正则表达式字符。主要的需要注意的“ . ”，“ * ”，和“ () ”。

本质上，“ . ”匹配任何字符，“ * ”匹配1个或者多个字符，“ () ”中的字符被记住用来给<replacement pattern>使用。

<replacement pattern>将产生用户的最后的验证DN。任何验证请求DN中匹配<search pattern>的“ () ”中的部分被保存在变量“ $1 ”。变量$1可以出现在<replacement pattern>中，并且将被验证请求DN中的字符串替换。如果<search pattern>中有多个“ () ”，将使用变量$2，$3，等等。

比如，假设用户的验证身份是如下所示的DN字符串：

```
uid=ADAMSON,cn=EXAMPLE.COM,cn=KERBEROS_V4,cn=AUTH
```

该用户的实际LDAP条目是：

```
uid=ADAMSON,ou=PERSON,dc=EXAMPLE,dc=COM
```

slapd.conf(5)中的saslRegexp指令应该如下所示：

```
saslRegexp
  uid=(.*),cn=example.com,cn=kerberos_v4,cn=auth
  uid=$1,ou=person,dc=example,dc=com
```

一个更加宽松的规则可以这样写：

```
saslRegexp

  uid=(.*),.*cn=auth

  uid=$1,ou=person,dc=example,dc=com
```

如果要设置更加宽松的搜索模式，一定要小心。因为可能会错误的允许个人作为一个他们不应该有访问权限的DN通过认证。最好写几个严格一点的指令，而不要写一个宽松的可能会有安全漏洞的指令。如果您的站点只有一种验证机制，并且不使用或者只使用一个域，您可能可以在验证身份和LDAP的DN之间使用一个单一的saslRegexp指令进行认证。

有一些站点将个人的DN扩展到LDAP树的多个区域，比如，有一个ou=accounting树和一个ou=engineering树，并且个人信息分布在它们之间。或者，在认证身份中没有足够的信息来区分DN，比如，如果上面的人员的LDAP条目如下所示：

```
dn: cn=mark adamson,ou=person,dc=example,dc=com

objectclass: Person

cn: mark adamson

uid: adamson
```

在这种情况下，认证身份中的信息只能被用来查找用户的DN，而不能直接从中产生用户的DN。对于这些情况以及其他的情况，saslRegexp指令中的<replacement pattern>需要用来产生一个LDAP URL，如下面的部分所描述的。

9.2.4、 为个人DN执行搜索

当在认证信息中没有足够的信息来直接产生个人的认证DN时，slapd.conf(5)中的saslRegexp指令将需要产生一个LDAP URL。该URL将被用来对LDAP内部的数据库执行搜索，以找到该人员的验证DN。

一个LDAP URL，和其他的URL相似，具有如下格式：

```
ldap://<host>/<base>?<attrs>?<scope>?<filter>
```

这包含了执行一个LDAP搜索所需要的所有元素：服务器的名称<host>，LDAP DN搜索的基<base>，用来获取LDAP属性的<attr>，搜索范围<scope>，<scope>可以使下面的3个选项之一：“ base ”，“ one ”或者“ sub ”，最后是LDAP的搜索过滤<filter>。因为该搜索是在本地机器上执行的LDAP DN搜索，<host>部分将被忽略。基于相同的原因，<attr>部分也被忽略，因为只有DN才是有用的。这两个元素在URL格式中保留是为了使得什么信息在字符串中的什么地方更

加清楚。

假设上面的例子中的个人的确有一个叫做“ adamson ”的认证用户名，并且该信息在LDAP条目的“ uid ”属性中保存。saslRegexp指令应该如下书写：

```
saslRegexp

 uid=(.*),cn=example.com,cn=kerberos_v4,cn=auth

 ldap://localhost/ou=person,dc=example,dc=com??sub?uid=$1
```

这将会初始化一个对于slapd服务器的LDAP数据库的内部搜索。如果该搜索返回正好一个条目，它被接受为该用户的DN，如果超过一个或者没有返回条目，认证将会失败，用户连接被作为认证请求DN保留。

注意，如果URL中的搜索范围<scope>是“ base ”，唯一返回的LDAP条目将是搜索数据库DN<base>，因此，实际上不会搜索数据库。这和上一部分中的直接将<replacement pattern>设置为DN的指令是等价的。

在URL中的搜索过滤<filter>中指定的属性应该被索引，以允许更快速的查找。如果没有，认证过程可能会很长，用户可能会认为服务器down机了。

9.3、 SASL授权

SASL提供了一个特性，叫做授权。它允许一个认证过的用户请求作为另外一个用户执行操作。该过程在用户得到一个认证DN之后执行，并且，包括向服务器发送一个授权身份。服务器将做出判断，允许或者拒绝授权发生。如果允许，该用户的LDAP连接切换到具有一个从授权身份中派生的绑定DN，LDAP会话以新的授权DN的访问权限继续。

判断允许一个授权是否继续，取决于运行LDAP的站点的规则和策略。并且不能单独由SASL完成。SASL库让服务器做出判断。LDAP管理员设置规则，通过在LDAP数据库条目中增加信息来确定谁可以授权给什么身份。

9.3.1、 授权的使用

该服务当一个实体需要以其他用户的身份执行操作时是非常有用的。比如，用户可能被定向到一个Web页面来修改他们在LDAP条目中的个人信息。用户向Web服务器进行认证来建立他们的身份。但是，Web服务器CGI不能作为该用户向LDAP服务器进行认证，以便修改改变。作为替代，Web服务器可以向LDAP服务器作为一个服务身份进行认证。比如：

```
cn=WebUpdate,dc=example,dc=com
```

然后，它将授权作为该用户的DN。授权之后，CGI将用户的LDAP条目做出改变，只要slapd服务器可以通过ACL。在连接的另外一端，是用户自己。用户本来可以以自己的身份直接连接到LDAP服务器，但是，这将需要用户具有更多有关LDAP客户端的知识。而Web页面提供了一个更简单的格式。

授权同样可以被用来限制一个对数据库具有更高访问权限的账户。该帐户甚至可以是在slapd.conf(5)中指定的root DN，它具有一个严格的个人列表，该列表可以授权为该DN。为了成为该DN，用户必须首先作为列表中的人员进行认证。这可以将谁对LDAP数据库进行了更新做出更好的审计。如果个人被允许直接认证为特权账户，很可能是直接通过slapd.conf(5)中的rootpw，或者通过一个userPassword属性，审计将会比较困难。

注意，在一个成功的授权之后，原始的LDAP连接中的认证DN被新的授权请求中的DN所替换。如果一个服务程序可以作为它自己的认证DN进行认证，然后授权作为其他的DN，并且，它还将在一个LDAP会话中切换到多个不同的身份，它将需要在每次授权作为其他DN的时候验证自己。slapd服务器不会保留服务程序切换到其他DN的能力的记录。在象Kerberos认证机制之类的机制上，这并不需要向Kerberos服务器进行多个连接，因为用户的TGT和“ ldap ”会话密钥在ticket的几个小时的生存期内可以被多次合法使用。

9.3.2、 为身份授权

授权身份通过ldapsearch的-X开关或者其他的工具发送给slapd服务器。或者是通过lutil_sasl_defaults()函数调用的*authzid参数。授权身份可以是以下两种格式之一，或者是：

```
u:<username>
```

或者：

```
dn:<dn>
```

在第一种格式中，<username>是来自于和上面的认证身份相同的名字空间。它是被底层认证机制引用的用户名。这种格式的认证身份被使用和认证过程相同的函数转换成一个DN格式。产生出如下所示的认证请求DN格式：

```
uid=<username>,cn=<realm>,cn=auth
```

授权请求DN然后被使用相同的saslRegexp过程，转换成数据库中的一个合法的授权DN。如果由于一个错误的LDAP URL搜索而导致不能转换，授权请求将因为“ 不适当的访问 ”而失败。否则，DN字符串现在是一个合法的授权DN格式，等待批准。

如果授权DN以第二种格式给出，使用一个{EX:"dn:"}}前缀，前缀之后的字符串已经是一个合法的授权请求DN，等待批准。

9.3.3、 授权规则

一旦slapd获得了授权DN，就开始了真正的批准过程。LDAP管理员可以将两个属性放到LDAP条目中来允许授权：

saslAuthzTo

saslAuthzFrom

两个属性都可以是多值的。第一个称为源规则。它被放到个人认证DN条目中，来说明该个人可以被允许切换到的授权DN。第二个被称为目的规则，它被放到个人一个授权DN条目中，来说明要切换到该授权DN，个人必须来自于哪一个认证DN。选择哪种方法取决于系统管理员。源规则在个人的认证DN条目中首先被检查。如果没有指明允许授权的saslAuthzTo，则检查在授权DN条目中saslAuthzTo规则。如果没有指明任何授权规则，请求将被拒绝。返回“ 不适当的访问 ”。因为默认的行为是拒绝授权请求，因此只能指明允许的规则；没有相反的指明拒绝授权的规则。

在这两个属性中的值和saslRegexp指令的<replacement pattern>输出具有相同的格式：或者是一个DN，或者是一个LDAP URL。比如，如果一个saslAuthzTo的值是一个DN，该DN就是用户可以授权为的DN；另外，如果saslAuthzTo的值是一个LDAP URL，该URL被用来执行内部LDAP数据库的搜索，然后，认证过的用户可以授权为搜索过程返回的任何DN。如果一个LDAP条目如下所示：

```
dn: cn=WebUpdate,dc=example,dc=com

saslAuthzTo: ldap://host/dc=example,dc=com??sub?objectclass=Person
```

那么，任何作为cn=WebUpdate,dc=example,dc=com认证的用户将可以授权为在搜索根dc=example,dc=com下，并且是具有“ Person ”对象的任何其他的LDAP条目。

9.3.3.1、 授权规则中的注意事项

在一个saslAuthzTo或者saslAuthzFrom属性中的一个LDAP URL将返回一组DN。每一个返回的DN将会被检查。返回大量的DN集合将导致授权过程消耗很长的时间。同样，应该在slapd创建了索引的属性上执行搜索。

为了让saslAuthzTo和saslAuthzFrom产生更加大量的规则，这两个属性的值可以是包含正则表达式字符的DN。这意味着如下所示的源规则：

```
saslAuthzTo: uid=.*,dc=example,dc=com
```

将允许一个通过认证的用户授权为任何符合给出的正则表达式模式的DN。该正则表达式可以比一个“ uid=* ”的LDAP查询执行的快的多。

同时注意，在一个授权规则中的值必须是两种格式之一：一个LDAP URL或者是一个DN（使用或者不使用正则表达式字符）。任何不以“ ldap:// ”开头的东西都将作为DN。不可能输入另外一个格式是“ u:<username> ”的授权身份作为一个授权规则。

确定使用哪一种规则，saslAuthzTo或者saslAuthzFrom，取决于站点的情况。比如，如果可以授权为一个给定身份的一组用户可以很简单的通过书写一个搜索过滤来表示，应该使用一个简单的目的规则；如果一组用户不能简单的被一个搜索过滤来定义，并且该组用户的数量很小，更好的方法是，在那些应该被允许授权的个人的条目中书写一个源规则。

10、 使用TLS

OpenLDAP客户端和服务器可以使用传输层安全TLS框架来提供完整性和机密性保护。并且通过SASL EXTERNAL来支持LDAP认证。

11、 构建分布式目录服务

对于许多站点而言，运行一个或者多个slapd(8)服务器，它们保存了整个的数据树，就足够了。但是，经常需要让一个slapd对于特定的树的部分引用其他的目录服务（可能运行或者不运行slapd）。

slapd支持下级和上级知识信息。

11.1、 下级知识信息

下级知识信息可以被用来代表一个子树。下级知识信息在目录中的代理点上作为一个特殊的引用对象被保存。该引用对象作为一个代理点来操作，将两个服务联系在一起。这种机制允许创建一个层次型的目录服务。

一个引用对象具有一个referral结构型的对象类，并且具有和被代理的子树相同的DN。通常，引用对象将提供一个辅助对象类extensibleObject。这允许一个条目包含正确的RDN数值。这最好通过一个例子来说明。

假设服务器a.example.com保存了dc=example,dc=net，并且希望将子树ou=subtree,dc=example,dc=net用另外一个服务器b.example.net代理，下面的命名引用对象应该被增加到a.example.net：

```
dn: dc=subtree,dc=example,dc=net

objectClass: referral
```

```
objectClass: extensibleObject

dc: subtree

ref: ldap://b.example.net/dc=subtree,dc=example,dc=net
```

该服务器通过这些信息来产生引用，并且搜索将在下级服务器上继续进行。

对于熟悉X.500的用户而言，一个命名引用对象和X.500中在subr DSE中保存的知识引用是很相似的。

11.2、 上级知识信息

上级知识信息可以使用referral指令说明。其值是一个引用到上级目录服务的URL列表。对于没有直接上级的服务器，比如上面的示例中的a.example.net，服务器可以配置为使用全局知识的目录服务，比如，OpenLDAP的根服务（http://www.openldap.org/faq/index.cgi?file=393）。

```
referral      ldap://root.openldap.org/
```

但是，因为a.example.net是b.example.net的直接上级，b.example.net应该被配置如下：

```
referral      ldap://a.example.net/
```

服务器使用这些信息来对于对如下信息的操作产生操作的引用：不在数据库中保存的名字空间中的信息；或者是名字空间中作为下级的信息。

对于熟悉X.500的用户而言，一个命名引用对象和X.500中在supr DSE中保存的知识引用是很相似的。

11.3、 ManageDsaIT控制

增加，修改和删除引用对象通常是使用ldapmodify(1)或者类似的支持ManageDsaIT控制的工具实现。ManageDsaIT控制通知服务器，你需要将引用object作为一个正常的条目来修改。这阻止服务器对于那些访问或者修改引用对象的请求发送一个引用结果。ldapmodify(1)的-M参数允许ManageDsaIT。例如：

```
ldapmodify -M -f referral.ldif -x -D "cn=Manag,dc=example,dc=net" -W
```

或者使用ldapsearch(1)：

```
ldapsearch -M -b "dc=example,dc=net" -x "(objectclass=referral)" '*' ref
```

注意：**ref**属性是可操作的，并且如果在搜索结果中需要，必须明确请求。

12、 使用SLURPD进行复制

在特定的配置中，一个单一的slapd(8)实例可能对于大量的通过LDAP来请求目录服务的客户端是不够的。运行一个或者多个slapd实例可能是必须的。在很多站点上，比如，有多个slapd服务器：一个主服务器和一个或者多个附属服务器。可以设置DNS来将对ldap.example.com的查询返回这些服务器的地址，在它们中间分布负载（或者只是在附属服务器之间分布负载）。这种主/附属安排提供了一个简单有效的方法来提高服务容量，可用性和可靠性。

slurpd(8)提供了从主slapd向附属slapd实例传播改变的能力。如果实现了如上所述的主/附属复制模式，slurpd和主slapd实例在同一台机器上运行。

12.1、 简述

slurpd(8)提供了“ in band ”的复制服务。也就是说，它使用LDAP协议来从主数据库更新附属数据库。最简单的说明方法是通过一个示例。在该示例中，我们跟踪一个LDAP的修改操作的传播，从它被一个LDAP客户端初始化，到分布到附属slapd实例上。

示例复制场景：

¢ LDAP客户端向从属slapd提交一个LDAP修改请求；

¢ 从属slapd给LDAP客户端返回一个指向主slapd的引用；

¢ LDAP客户端向主slapd提交LDAP修改请求操作；

¢ 主slapd执行修改请求，将改变写入复制日志文件，然后给客户端返回一个成功代码；

¢ slurpd进程注意到一个新的条目附加到复制日志文件，读取该复制日志条目，然后通过LDAP将改变发送给从属slapd。

¢ 从属slapd执行修改操作，给slurpd进程返回一个正确的代码。

12.2、 复制日志

当slapd被配置为产生复制日志文件，它向一个包含LDIF改变记录的文件中写入。复制日志给出了复制站点，一个时间戳，被修改的DN的条目，以及指明了所发生的改变的多行信息。在下面的示例中，Barnara(uid=bjensen)替换了description的值。这种改变被传播到运行在slave.example.com

的从属slapd实例。对多个操作属性的改变，比如modifiersName以及modifyTimestamp，被包含在改变记录中，并且将被传播到从属slapd。

```
replica: slave.example.com:389

time: 809618633

dn: uid=bjensen,dc=example,dc=com

changetype: modify

replace: multiLineDescription

description: A dreamer...

-

replace: modifiersName

modifiersName: uid=bjensen,dc=example,dc=com

-

replace: modifyTimestamp

modifyTimestamp: 20000805073308Z
```

对modifiersName以及modifyTimestamp操作属性的改变被主slapd服务器增加。

12.3、 命令行选项

该部分详细说明了常用的slurpd(8)的命令行选项：

```
-d <level> | ?
```

该选项设置slurpd的调试级别为<level>。当<level>是一个“？”字符时，打印出不同的调试级别的信息，slurpd终止，无论您给了它其他的任何参数选项。当前的调试级别有（slapd的调试级别的子集）：

Table 12.1: Debugging Levels

Level	**Description**
4	heavy trace debugging

64	configuration file processing
65535	enable all debugging

调试级别是可以相加的。因此，如果您想指定heavy trace debugging，并且想看到正在处理的配置文件，您可以这顶着两个调试级别的和（在本示例中是68）。

-f <filename>

该选项指明了一个另外的slapd配置文件。slurpd没有自己配制文件，所有的配置信息都是从slapd的配置文件中读取的。

-r <filename>

该选项指明了另外的slap复制日志文件。在正常情况下，slurpd从slapd的配置文件中读取复制日志文件的名称。但是，您可以使用-r标志覆盖该选项，以便让slurpd处理一个不同的复制日志文件。请参阅“ 高级slurpd操作 ”部分来查看关于如何使用此选项的讨论。

-o

在“ one-shot ”模式下操作。正常情况下，当slurpd结束处理一个复制日志，它将保持活动，并且周期性的检查，是否有新的条目被增加到复制日志。在“ one-shot ”模式下，slurpd处理一个复制日志文件，并且马上终止。如果给出了-o选项，复制日志文件必须被明确使用-r选项说明。请参阅“ One-shot模式和拒绝文件 ”部分来获得对于该选项的讨论。

-t <directory>

说明一个另外的目录作为slurpd的临时目录，用来拷贝复制日志。缺省的位置是/usr/tmp。

12.4、 配置slurpd和从属slapd实例

为了运行一个复制slapd实例，必须为复制配置主slapd和从属slapd实例。然后，关掉主slapd实例以便拷贝数据库。最后，运行主slapd实例，从属slapd实例，和slurpd实例。这些步骤在下面的部分详细描述。只要你愿意，可以设置任意多的从属slapd实例。

12.4.1、 设置主*slapd*

下面的部分假设您已经有一个正在运行的slapd实例。为了配置您的正在运行的slapd服务器作为一个主要的复制服务器，必须在slapd.conf中进行下面的改变。

¢ 为每一个复本增加一个replica指令。参数binddn=应该和对应的从属slapd配置文件的updatedn选项相

匹配，并且，应该命名一个对于从属数据库具有写全权限的条目（比如，作为rootdn列出的条目，或者通过从属slapd配置文件的access指令允许访问）。

¢ 增加一个replogfile指令，该指令告诉slapd在何处记录改变。该文件将被slurpd读取。

12.4.2、 设置从属*slapd*

在一个要作为从属slapd服务器的机器上安装slapd软件。该从属服务器的配置文件应该和主服务器的配置文件相同，除了下面的部分：

¢ 不要包含replica指令。因为它可能会包含复制链。在大多数情况下是不是不适当的。

¢ 不要包含replogfile指令。

¢ 确认包含一个updatedn行。在该处给出的DN应该和相关的主slapd的配置文件中的replica=指令给出的binddn=参数指明的DN匹配。

¢ 使用updateref指令定义一个URL，当从属slapd接收到一个更新请求时将返回该URL。

12.4.3、 关闭主*slapd*

为了确保从属slapd包含了主slapd的确切的拷贝，必须停止主slapd。通过使用kill-INT <pid>给主slapd发送一个中断信号来停止主slapd服务器。<pid>是主slapd进程的进程ID。

如果愿意，您还可以在复制数据库的时候重新在只读状态下启动slapd服务器。这时，如果客户端试图修改数据，主slapd将返回一个“ unwilling to perform ”错误。

12.4.4、 将主slapd的数据库拷贝到从属slapd

将朱服务器的数据库拷贝到从属服务器。对于一个LDBM数据库，必须拷贝在slapd.conf(5)文件中的directory指明的数据库目录下的所有的数据库文件。数据库文件根据使用的底层数据库的不同具有不同的后缀。当前可能的后缀有：

Table 12.2: Database File Suffixes

Suffix	**Database**
dbb	Berkeley DB B-tree backend
dbh	Berkeley DB hash backend
gdbm	GNU DBM backend

通常，应该拷贝在数据库目录下的所有文件，除非您能够确保它没有被slapd(8)所使用。

注意：拷贝过程假设服务器是同构的，安装配置了相同的**OpenLDAP**。

12.4.5、 为复制设置主slapd

为了配置slapd产生一个复制日志文件，在主slapd的配置文件中增加一个“ replica ”配置选项。比如，如果我们希望将改变传播到运行在服务器slave.example.com的slapd实例：

```
replica host=slave.example.com:389
        binddn="cn=Replicator,dc=example,dc=com"
        bindmethod=simple credentials=secret
```

在这个示例中，更新将被发送给slave.example.com的389（标准LDAP端口）。slrupd进程将作为"cn=Replicator,dc=example,dc=com"向从属slapd进行绑定，使用简单认证机制和口令secret。注意，由binddn=指令指定的DN必须在从属slapd的数据库中存在，或者是slapd配置文件中说明的rootdn，只有这样，绑定操作才会成功。该DN同时必须在从属服务器的sladp.conf(5)文件中作为updatedn列出。

注意：高度推荐使用高强度的认证和传输层加密！

12.4.6、 重新启动主slapd并且启动从属slapd

重新启动主slapd进程。为了检查slurpd能够产生复制日志，对数据库中的任何条目执行一个修改操作，然后检查数据已经被写入到日志文件。

12.4.7、 启动slurpd

启动slurpd进程。slurpd应该立即将测试的修改复制到从属slapd服务器。检查从属slapd的日志文件来确认修改已经发送。

```
slurpd -f <masterslapdconfigfile>
```

12.5、 高级slurpd操作

12.5.1、 复制错误

当slurpd向从属slapd传播改变的时候，如果接收到一个错误的返回代码，它将错误原因和复制记

录写入到一个拒绝日志文件。该文件和每一个复制日志文件位于相同的目录，并且具有相同的名称，只是附加了“ .rej ”字符串。比如，对于运行在slave.example.com，端口389的服务器，拒绝日志文件如果存在的话，将被命名为：

```
/usr/local/var/openldap/replog.slave.example.com:389.rej
```

一个示例拒绝日志文件的条目如下：

```
ERROR: No such attribute

replica: slave.example.com:389

time: 809618633

dn: uid=bjensen,dc=example,dc=com

changetype: modify

replace: description

description: A dreamer...

-

replace: modifiersName

modifiersName: uid=bjensen,dc=example,dc=com

      -

replace: modifyTimestamp

modifyTimestamp: 20000805073308Z
```

注意，该文件的格式和原始的复制日志条目具有完全相同的条目，只是在条目的前面增加了一个ERROR行。

12.5.2、 One-shot模式和拒绝文件

使用slurpd的“ One-shot模式 ”来处理一个拒绝日志文件是可能的。在正常的操作下，slurpd检测被附加到日志文件中的更多的复制记录。作为比较，在One-shot模式下，slurpd处理一个单一的日志文件，然后终止。slurpd忽略复制日志文件条目开始处的ERROR行，因此，在将它传递给拒绝文件日志之前，不需要编辑它们

为了使用One-shot模式，在命令行中将拒绝日志文件的名称作为-r标志的参数，然后，使用-o标志指明One-shot模式。比如，为了处理一个拒绝日志文件/usr/local/var/openldap/replog.slave.example.com:389然后终止，使用下面的命令：

```
slurpd -r /usr/tmp/replog.slave.example.com:389 –o
```

13、 附录A 常用配置指令

基本安装

下面是基本安装指南。

“ configure ”SHELL脚本试图猜测出编译时使用的正确的系统相关的变量。然后，使用这些值在每一个软件包的目录中创建一个“ Makefile ”。它同时创建一个或者多个包含系统相关的定义的“ .h ”文件。最后，它创建一个“ config.status ”SHELL脚本，您可以在将来运行该脚本来重新创建当前的配置。一个“ config.cache ”文件包含了检测到的结果，用来加速重新配置过程。还有一个文件“ config.log ”，包含了编译输出（主要用来调试“ configure ”）。

如果需要编译软件包来做一些不常见的功能，请仔细阅读“ configure ”是否能够检查，并且将不同之处（diffs）或者指令发送到README中指出的地址，以便在下一个版本中考虑。如果在“ config.cache ”文件中包含了您不需要的信息，可以删除或者编辑它。

“ configure.in ”文件被用来被一个叫做“ autoconf ”的程序创建“ configure ”。如果你需要用新版本的“ autoconf ”来改变或者重新产生“ configure ”，您只需要“ configure.in ”文件。

最简单的编译软件包的方法是：

¢ “ cd ”到包含软件包源代码的目录，输入“ ./configure ”来为您的系统配置软件包。如果您使用旧版本的SystemV上的“ csh ”，您可能需要输入“ sh ./configure ”来阻止“ csh ”试图自己执行“ configure ”；

¢ 输入“ make ”来编译软件包；

¢ 作为可选项，可以输入“ make check ”来执行随软件包提供的测试；

¢ 执行“ make install ”来安装程序和数据文件，以及文档；

¢ 可以输入“ make clean ”来从源代码目录删除二进制程序代码和目标文件。输入“ make distclean ”来同时删除“ configure ”创建的文件（以便您可以在不同的机器上编译软件包），还有一个

叫做“ make maintainer-clean ”的MAKE TARGET，但是，该选项主要是为软件包的开发者使用的。如果您使用它，为了创新产生出随软件包一起发行的文件，您必须获得所有的其他程序。

编译器和选项

某些系统对于编译和连接需要一些“ configure ”代码不知道的特殊选项。您可以通过在环境变量中给“ congfigure ”设置变量的方法给“ configure ”传递初始化值。使用一个和Bourne SHELL兼容的SHELL，可以如下所示在命令行中进行设置：

```
CC=c89 CFLAGS=-O2 LIBS=-lposix ./configure
```

或者在一些具有“ env ”程序的系统中，可以如下所示进行设置：

```
env CPPFLAGS=-I/usr/local/include LDFLAGS=-s ./configure
```

为多种体系结构编译

您可以通过将每一种体系结构的目标文件放置到它们自己的目录中，来同时为多于一种计算机编译软件包。要支持这种特性，必须使用支持“ VPATH ”变量的“ make ”版本。比如DNU“ make ”。首先，“ cd ”到需要将目标文件和可执行文件放置到的目录，然后运行“ configure ”脚本。“ configure ”自动检测它所在的目录中的源代码，以及在“ .. ”目录中的源代码。

如果您必须使用不支持“ VPATH ”环境变量的“ make ”，必须针对每一种体系结构在源代码目录中编译一次。在安装了一种体系结构的软件包之后，在重新配置为另外一种体系结构之前，使用“ make disclean ”命令。

安装名称

默认情况下，“ make install ”将在“ /usr/localbin ”，“ /usr/local/man ”等目录下安装软件包的文件。您可以使用“ configure ”的“ --prefix=PATH ”选项指明一个其他不同于“ /usr/local ”的安装前缀。

您可以为体系结构相关的文件和体系结构无关的文件指定分别的安装前缀。如果您为“ configure ”指明了“ --exec-prefix=PATH ”选项，软件包将使用PATH作为前缀来安装程序和库文件。文档和其他的数据文件仍然使用正常的前缀。

另外，如果您使用了非正常的目录结构，您可以指定象“ —binddir=PATH ”这样的选项来为特斯农工的文件类型指明不同的值。运行“ configure --help ”来查看您可以设置的目录列表，以及什么类型的文件将被放到该目录中。

如果软件包支持，您可以让程序在它们的名字中使用特别的前缀或者后缀来安装。这可以通过

为“ configure ”指定“ --program-prefix=PREFIX ”或者“ —program-suffix=SUFFIX ”来实现。

可选的特性

某些软件包特别注意“ configure ”中的“ --enable-FEATURE ”选项，其中，FEATURE指明了软件包的可选部分。它们也可能注意“ --with-PACKAGE ”选项，其中，PACKAGE在某种程度上和“ gnu-as ”或者“ x ”（对于X Window系统）很相似。README中提到了软件包能够识别的任何“ --enable ”或者“ --with- ”选项。

对于使用X Window系统的软件包而言，“ configure ”通常情况下能够自动发现X包含和库文件，但是如果它不能，可以为“ configure ”指明“ --x-includes=DIR ”和“ --x-libraries=DIR ”来指明它们的位置。

指明系统的类型

可能有一些“ configure ”不能自动识别的特性。但是需要根据软件包运行的主机系统来确定。通常情况下，“ configure ”能够识别出来，但是，如果它打印了一条信息说明它不能猜测出系统的类型，为它使用“ --host=TYPE ”选项。TYPE可以是系统的缩写名称，比如“ sun4 ”，或者是具有3个域的规范名称：

CPU-COMPANY-SYSTEM

查看文件“ config.sub ”来获得每一个域的可能的值。如果“ config.sub ”没有包含在软件包中，那么该软件包不需要知道系统类型。

如果您正在创建交叉编译的编译器，还可以使用“ --target=TYPE ”选项来选择将要产生的代码的系统类型，以及使用“ --build=TYPE ”来选择您正在其上编译软件包的系统类型。

共享默认值

如果您需要为共享（发行）“ configure ”脚本设置缺省的值，可以创建一个站点SHELL脚本，叫做“ config.site ”，并且为“ CC ”，“ cache_file ”和“ prefix ”等环境变量设置默认值。“ configure ”将查找“ PREFIX/share/config.site ”,如果它存在的话。然后，查找“ PREFIX/etc/config.site ”，如果它存在。或者，您可以设置“ CONFIG_SITE ”环境变量到该站点脚本的位置。警告：不是所有的“ configure ”脚本都查找一个站点脚本。

操作控制

“ configure ”识别下面的选项，来控制它是如何操作的。

“ --cache-file=FILE ”

将测试的结果使用并且保存到一个文件，而不是“ ./config.cache ”。将FILE设置为“ /dev/null ”将为调试“ configure ”而禁止缓存。

“ --help ”

打印一个“ configure ”的选项概要，然后终止。

“ --quiet ”

“ --silent ”

“ -q ”

不打印正在进行什么检查的信息。为了压缩正常的输出，可以将它重定向到“ /dev/null ”（任何错误信息还是能够显示）。

“ --srcdir=DIR ”

在目录DIR中查找软件包的源代码。通常，“ configure ”可以自动决定目录。

“ --version ”

打印产生“ configure ”脚本所使用的Autoconf的版本，然后终止。

“ configure ”同样还支持一些其他的不经常使用的选项。

14、 OpenLDAP软件版权

14.1、 OpenLDAP Copyright Notice

additional restrictions.

This work is derived from the University of Michigan LDAP v3.3 distribution. Information concerning this software is available at <http://www.umich.edu/~dirsvcs/ldap/>.

This work also contains materials derived from public sources.

Additional information about OpenLDAP software can be obtained at <http://www.OpenLDAP.org/>.

14.2、 University of Michigan Copyright Notice

Portions Copyright 1992-1996 Regents of the University of Michigan. All rights reserved.

Redistribution and use in source and binary forms are permitted provided that this notice is preserved and that due credit is given to the University of Michigan at Ann Arbor. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. This software is provided ``as is'' without express or implied warranty.

15、 The OpenLDAP Public License

The OpenLDAP Public License

Version 2.7, 7 September 2001

Redistribution and use of this software and associated documentation

("Software"), with or without modification, are permitted provided

that the following conditions are met:

1. Redistributions of source code must retain copyright statements

and notices,

2. Redistributions in binary form must reproduce applicable copyright

statements and notices, this list of conditions, and the following

disclaimer in the documentation and/or other materials provided

with the distribution, and

3. Redistributions must contain a verbatim copy of this document.

The OpenLDAP Foundation may revise this license from time to time.

Each revision is distinguished by a version number. You may use

this Software under terms of this license revision or under the

terms of any subsequent revision of the license.

THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND ITS

CONTRIBUTORS ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES,

INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY

AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT

SHALL THE OPENLDAP FOUNDATION, ITS CONTRIBUTORS, OR THE AUTHOR(S)

OR OWNER(S) OF THE SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT,

INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,

BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT

LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN

ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

POSSIBILITY OF SUCH DAMAGE.

The names of the authors and copyright holders must not be used in