a *gdbm* file hierarchy. To use it, type `tree2dbm database-directory` where database-directory is the directory where the EDB file hierarchy begins. This script does a find starting in the specified directory for files named EDB and runs them through the *edb2dbm* program which creates a file called *EDB.gdbm*. The original EDB file is neither removed nor molested, so you'll need roughly twice the disk space. Alternatively, you can run *edb2dbm* by hand on each EDB file. The reverse operation (converting from a *gdbm* hierarchy to an EDB hierarchy) is done by the *synctree* shell script. It is a good idea to run *synctree* out of *crontab* once in a while to update the EDB file hierarchy.

Finally, it should be noted that parse errors will be reported somewhat differently in the *dsap.log* file with the TURBO_DISK option enabled. Since line numbers do not make much sense in a *gdbm* file, errors will be reported based on the Relative Distinguished Name of the offending entry. If you get a parse error (because of a non-printable character, for example), the best approach is to do something like this:

```
edbcat EDB >bob
```

Edit the file *bob*, locate the entry, fix the problem, then

```
edb2dbm bob
mv bob.gdbm EDB.gdbm
```

where this procedure assumes you are in the directory containing the bad EDB.gdbm file. The *edbcat* program can be found in the *quipu/turbo* directory and is used to convert from *gdbm* back to plain text EDB format.

12.3 Files

Regardless of how you install QUIPU and the ISODE, the number of files needed to run QUIPU is quite small.

In ISODE's `BINDIR` directory, typically */usr/local/bin/*, there are a few programs of interest:

`dish`: The directory shell
This is discussed in Chapter 4.

`bind`: Shell interface to DISH
: There are actually several links (listed below) to a program called *bind*. These act to export the DISH interface to the UNIX shell. As such, you can issue commands to DISH from the shell, rather than running DISH directly.

```
add         compare
delete      dsacontrol
list        modify
modifyrdn   moveto
search      showentry
showname    squid
```

`fred`: A white pages user interface
: See Chapter 7.

`editentry`: Edit a directory entry
: This is a simple shell script that DISH invokes when you ask DISH to edit an entry in the directory.

`unbind`: Unbind from DISH
: This command is used to terminate DISH.

In ISODE's `SBINDIR` directory, typically */usr/etc/*, the DSA resides:

`ros.quipu`: The QUIPU DSA
: This program will be started once, for each DSA you are running, from rc.local. A script is provided to invoke this program, in case you need to restart it.

In ISODE's `ETCDIR` directory, also typically */usr/etc/*, there are a few programs and files of interest:

`oidtable.at`, `oidtable.gen`, `oidtable.oc`: These define the attribute types, generic object identifiers, and object classes known to the system. (An object identifier is a method used to unambiguously encode, among other things, the names of attributes and object classes.) These files you never deal with unless they are accidentally corrupted.

`dsaptailor`: This is the run-time tailor file for the DUAs. You will configure this file initially and then probably leave it alone.

`isoaliases`, `isobjects`, `isoentities`, `isomacros`, `isoservices`: These are various databases used by the ISODE. These files you never deal with unless they are accidentally corrupted.

`isologs`: This script runs nightly under *cron*(8) to trim the ISODE log files, kept in ISODE's `LOGDIR` directory, typically */usr/tmp/*. This file you never deal with unless it is accidentally corrupted.

`isotailor`: This is the run-time tailor file for the ISODE. You will configure this file initially and then probably leave it alone.

Chapter 13

Configuring a DUA

It is suggested that you try to get a DUA operational by connecting to a "well-known" DSA before you attempt to operate a local DSA. Or, if you have a DSA from a previous release of QUIPU — try and connect to that.

13.1 Connecting to a DSA

A DUA essentially only needs to know one thing to be able to contact a DSA: that is the OSI network address of the DSA. This parameter is defined in the file *dsaptailor*, together with some other parameters. The full set of parameters is described in Section 13.2.

The `dsa_address` parameter defines a local name and the network address of the DSA to initially contact. For example,

```
dsa_address  vicuna  Internet=bells.cs.ucl.ac.uk+50987
```

declares that the DSA locally referred to by the name `vicuna` is contacted by calling the network address `Internet=bells.cs.ucl.ac.uk+50987`. The syntax of network addresses is discussed in briefly in Section 14.2.1 and more fully in *Volume Two* of this manual and [Kil89a].

As shown, the address is preceded by a private key `vicuna`. This can be used in some DUAs (including DISH) to specifying the address of the DSA to contact. If there are more than one `dsa_address` entries, the first entry will be used to supply the default DSA address.

A default *dsaptailor* file (taken from the *dsap/dsaptailor* file of the ISODE source tree) is installed as *dsaptailor* in the ISODE `ETCDIR` directory (usually

/usr/etc/) when the *dsap* library is installed, this supplies the addresses of various DSAs that you may be able to access.

To try to connect to one of the DSAs listed in *dsaptailor*, invoke DISH, with a `-call` flag; e.g., `dish -call giant` will try to contact the DSA "giant" (Giant Tortoise!) running at the University of London Computer Centre (ULCC). If the connection is successful, then the prompt `Dish ->` will be returned. If the connection fails, the program will exit with an appropriate error message.

If this fails you might want to try connecting to some of the other registered DSAs, for example, try `dish -call alpaca`, `dish -call eel` or `dish -call anaconda`.

Many of these top level DSAs do not allow anonymous connections. If you see the message `inappropriate authentication` as a result of your connection attempt you will probably need to supply a DN using the `-username` flag to DISH.

If you fail to contact a DSA at this point, there are likely to be lower level problems. You should turn up the ISODE logging (see *Volume Two* of this manual) to see what is happening to the network calls.

If you invoke DISH without a `-c` flag (using the default *dsaptailor*), it will try to connect to the DSA defined by the first `dsa_address` entry.

DISH is described in full in Chapter 4 of this manual.

13.2 Tailoring

The program configuration is tailored to allow you to change logging levels, and other parameters at runtime. It is used by the QUIPU DUA procedures, and by the QUIPU User Interfaces.

The file *dsaptailor* is used for this purpose and consists of single value entries (e.g., oidtable), unless otherwise stated (e.g., dsaplog). Each entry has a parameter followed by its value. The various options are:

`oidtable:` The path for the OID definition tables. Note: It is best to have this appear as the first entry of the *tailor* file, as other entries may contain attributes that need to be looked up in these tables There are three:

1. *file.gen*, which contains generic names for building OIDs;

2. *file*.at, which contains the OIDs for attributes; and,
3. *file*.oc, which contains the OIDs for object classes.

For example,

```
oidtable    /usr/lib/quipu/OIDTable
```

will direct the DSA to consult:

```
/usr/lib/quipu/OIDTable.gen
/usr/lib/quipu/OIDTable.at
/usr/lib/quipu/OIDTable.oc
```

By default this variable is set to `oidtable` which refers to the tables *oidtable.** in the ISODE `ETCDIR` directory.

`dsa_address:` This parameter is described in Section 13.1.

`dsaplog:` Tailoring for the normal logging file. Each entry consists of one or more key/value pairs expressed as:

```
key=value
```

The keys are:

`file:` The name of the logfile.

`size:` The size in KBytes to which the logfile should be allowed to grow. When the log has reached this size, if the "zero" option below is set, then the log will be truncated; otherwise, no further logging will take place.

`level:` The levels of logging to be written to this log file. This can be any of the following levels:

`fatal:` fatal errors only.

`exceptions:` serious, but hopefully temporary, errors.

`notice:` general logging information.

`trace:` program tracing.

`pdus:` pdu tracing.

`debug:` full tracing of events.

`all:` log all events.

For example to have all errors written to the file you will need

```
dsaplog level=fatal level=exceptions
```

`dlevel:` Do not log the specified log level, this is the opposite of the above entry.

`dflags/sflags:` The flags associated with the log may be set (with `sflag`) or unset (with `dflag`). The allowable options are:

`close:` close the log after each entry.

`create:` create the log file if it does not exist.

`zero:` truncate the file when it gets too big.

`tty:` copy the logging information to the users tty.

An example might be:

```
dsaplog level=notice size=30 file=quipulog dflags=close
```

This says log events at "notice" level into the file *quipulog*, do not let the file grow larger than 30Kbytes; and do not close the file after each logging message.

`stats:` Used to control the level of statistical logging (parameters as for `dsaplog` above).

`local_DIT:` The argument is a distinguished name. When some User Interfaces start, you will be automatically moved to this position in the DIT.

`oidformat:` Defines how object identifiers should be printed. Use

```
oidformat   short
```

to print in short local key form, e.g.,

```
Country
```

or,

```
oidformat   long
```

to print in long object identifier form, e.g.,

```
joint.ds.attributeType.country
```

or,

```
oidformat   numeric
```

to print in numeric form, e.g.,

```
2.5.4.6
```

`photo:` The argument has two parts, a "terminal type" such as "sun" or "xterm"; the second is the name of the process that should handle displaying of photographs; for example,

photo xterm Xphoto

tells the DUA to call the process Xphoto to handle photograph attributes if the user is on a terminal of type "xterm". Handling photographs is described more fully in Section 14.7.3.

`quipurc:` If the argument has the value `on`, then a program called *dishinit* will be run every time a user without a *.quipurc* file tries to access the directory. *dishinit* is discussed in Section 4.7.1.

`sizelimit:` Defines the maximum number of entries a successful list or search should return. For example,

```
sizelimit 20
```

sets the DAP default service control "sizelimit" to be 20 entries.

`timelimit:` Defines the maximum number of seconds a list or search should return. For example,

```
timelimit 30
```

sets the DAP default service control "timelimit" to be 30 seconds.

`ch_set:` If set to `ISO8859` this tells a DUA you have access to an ISO 8859-1 font which can be used for displaying T.61 characters, see Section 10.1.2 for details.

Chapter 14

Configuring a DSA

This chapter discusses how to configure a QUIPU DSA. We recommend that you get a DUA running before you try to get a DSA working.

14.1 Basic Formats and Structures

All of the information a DSA requires is stored on disk and is text-structured. This includes various files (described later), and the local DIT database itself. A complete BNF description of the files is given in Appendix B on page 257.

14.1.1 Entry Data Block

A key component of the directory database is the entry data block, which is described fully in [Kil89b]. Figure 14.1 shows an example EDB file containing two `person` entries.

An EDB file contains a header, this is optionally but typically followed by a sequence of entries.

The header consists of two lines of text, the first must contain the string `MASTER`, `SLAVE` or `CACHE`, which indicates whether the data in the EDB file represents the authoritative `master` data, a `slave` copy of all the data, or some `cached` entries.

The next line of the EDB is a string that describes the version of the EDB. Every time the EDB is altered, the version number should be changed, so that `SLAVE` EDBs elsewhere will be automatically updated. When a DSA

```
MASTER
19891025113003Z
CN= Colin Robbins
CN= Colin John Robbins
Phone= +44-1-387-7050 ext 3683
Surname= Robbins
Room= G10
Userid= crobbins
userClass= csstaff
rfc822Mailbox= C.Robbins@cs.ucl.ac.uk
Photo= {FILE}crobbins.photo
objectClass =thornPerson & quipuObject
acl=others # none # attribute # photo
acl=self # read # attribute # photo

CN= Steve Kille
CN= Steve E. Kille & Stephen Kille
Phone= +44-1-387-7050 ext 7294
Surname= Kille
objectClass = thornPerson & quipuObject
Room= G24
Userid= steve
userClass= csstaff
rfc822Mailbox= S.Kille@cs.ucl.ac.uk
```

Figure 14.1: Example EDB File