unixODBC without the GUI

Or
everything you wanted to know about odbcinst but were afraid to ask

Purpose

A lot of people are using unixODBC but for a number of reasons are not building the GUI configuration and testing tools (ODBCConfig and DataManager). This document is aimed at these people and hopes to explain what you need to do and when to do it.

What's a ini file ?

ODBC first appeared within Windows 3.0. At this time Windows used .ini files to contain configuration information. These are text files containing the following layout

```
[section1]
entry1 = value
entry2 = value

[section2]
entry1 = value
entry2 = value
...
```

With the advent of Windows NT these ini files have been replaced by the registry, but the API to access them in ODBC has remained the same. Windows has two function in odbcinst.dll that allow applications and drivers to query and modify these files, SQLGetPrivateProfileString and SQLPutPrivateProfileString.

As part of unixODBC's aim of reproducing the ODBC environment on non Windows platform's the ini files and libodbcinst provide the same format and functionality.

System versus User

ODBC distingushes between two types of ini files. System ini files are designed to be accessable but not modifable by any user, and user files are private to a particular user, and may be modified by that user.

The system files are odbcinst.ini and odbc.ini (note no leading dot), and the user file is ~/.odbc.ini in each user's home directory (note leading dot).

The system file odbcinst.ini contains information about ODBC drivers available to all users, and the odbc.ini file contains information about DSN's available to all users. These "System DSN's" are useful for

application such as web servers that may not be running as a real user and so will not have a home directory to contain a .odbc.ini file.

A good example of this is Apache and PHP with ODBC support. When the http server is first started it calls SQLAllocEnv as root. it then at a later time changes to the specified user (in my case nobody) and calls SQLConnect. If the DSN's was not a system DSN then this fails.

FILEDSN's

ODBC 3 also has a third sort of DSN, a file DSN. These store the connection information in a file that may be accessable to anyone. unixODBC does not at this time support FILEDSN's but it will when I get around to it. They are useful things but of less use to UNIX's than NT. Because of the MS view that everyone should have Windows on their desk, each workstation will have it's own registry with it's own set of system and user DSN's that can not be used by other workstations. File DSN's are a fix to allow the information to be stored in a central server that is accessable to all the workstations.

Why not vi ?

All the configuration files needed by unixODBC are plain text files, so there is no reason that you can not use your favorite text editor to setup the files.

However since beta 1.6 the location of the system files odbcinst.ini and odbc.ini are determined by the configure script. The default location is /usr/local/etc, and if a prefix is specified the location is {prefix}/etc. The location of the etc path can be broken out of the normal prefix tree by specifing --sysconfdir=DIR, so the following will expect the system files to be in the same location as pre 1.6 builds.

```
./configure --sysconfdir=/etc
```

The upshot of all this is that if you use odbcinst to configure the files you can be sure that the same path to the files will be used as are used by the driver manager, so the modifications will take effect.

What goes into them ?

Ok now we know a bit of the history of ini files and ODBC so now we need to get to the bit that is actually of use. What you put in them.

odbcinst.ini

This contains a section heading that provides a name for the driver, so for the example below PostgreSQL to indicate a Postgres driver. The following lines contain a description and then the important bits. The Driver and Setup paths point to the ODBC driver and setup libs. The setup lib is used when you click on Add in ODBCConfig to add a new DSN, but as this document is about not using the GUI tools, this is not that important for us. Far more important is the Driver entry (vital in fact) This is the library that the driver manager will dynamicaly load when SQLConnect or SQLDriverConnect is called for that DSN. If this points to the wrong place the DSN will not work. If the dlopen() fails the DSN will not work. The fileusage entry is added by the odbcinst program, so if you are using a text editor, you will need to add it yourself.

```
[PostgreSQL]
Description     = PostgreSQL driver for Linux & Win32
Driver          = /usr/local/lib/libodbcpsql.so
Setup           = /usr/local/lib/libodbcpsqlS.so
FileUsage       = 1
```

templates

odbcinst expects to be supplied with a template file. If you are adding a driver for the above entry the template file would contain the following

```
[PostgreSQL]
Description     = PostgreSQL driver for Linux & Win32
Driver          = /usr/local/lib/libodbcpsql.so
Setup           = /usr/local/lib/libodbcpsqlS.so
```

and you would invoke odbcinst with the following arguments, assuming that you have created a file template_file with the above entries in.

```
odbcinst -i -d -f template_file
```

The args to odbcinst are as follows

-i install
-d driver
-f name of template file

Threads

Since 1.6 if the driver manager was built with thread support you may add another entry to each driver entry. For example

```
[PostgreSQL]
Description     = PostgreSQL driver for Linux & Win32
Driver          = /usr/local/lib/libodbcpsql.so
Setup           = /usr/local/lib/libodbcpsqlS.so
Threading       = 2
```

This entry alters the default thread serialization level. More details can be found in the file DriverManager/__handles.c in the source tree.

[.]odbc.ini

The contents of the odbc.ini files are a bit more complicated, but they follow just the same format as the odbcinst.ini entries. These are complicated by each driver requiring different entries. The entries for all the drivers supplied with the distribution are included bellow for reference. The entries may be added in the same way using odbcinst, or a text editor. A sample entry to match the above driver could be

```
[PostgreSQL]
Description         = Test to Postgres
Driver              = PostgreSQL
Trace               = Yes
TraceFile           = sql.log
Database            = nick
Servername          = localhost
UserName            =
Password            =
Port                = 5432
Protocol            = 6.4
ReadOnly            = No
RowVersioning       = No
ShowSystemTables    = No
ShowOidColumn       = No
FakeOidIndex        = No
ConnSettings        =
```

And this may be written to a template file, and inserted in the ini file for the current user by

```
odbcinst -i -s -f template_file
```

The individual entries of course may vary.

The Driver line is used to match the [section] entry in the odbcinst.ini file and the the Driver line in the odbcinst file is used to find the path for the driver library, and this loaded and the connection is then established. It's possible to replace the driver entry with a path to the driver itself. This can be used, for example if the user can't get root access to setup anything in /etc (less important now because of the movable etc path). For example

```
[PostgreSQL]
Description         = Test to Postgres
Driver              = /usr/local/lib/libodbcpsql.so
Trace               = Yes
TraceFile           = sql.log
Database            = nick
Servername          = localhost
UserName            =
Password            =
Port                = 5432
Protocol            = 6.4
ReadOnly            = No
RowVersioning       = No
ShowSystemTables    = No
ShowOidColumn       = No
```

```
FakeOidIndex        = No
ConnSettings        =
```

Templates

The templates for the included drivers are...

PostgreSQL

```
[Postgres]
Description         = Test to Postgres
Driver              = PostgreSQL
Trace               = Yes
TraceFile           = sql.log
Database            = nick
Servername          = localhost
UserName            =
Password            =
Port                = 5432
Protocol            = 6.4
ReadOnly            = No
RowVersioning       = No
ShowSystemTables    = No
ShowOidColumn       = No
FakeOidIndex        = No
ConnSettings        =
```

Mini SQL

```
[Mini SQL]
Description     = MiniSQL
Driver          = MiniSQL
Trace           = No
TraceFile       =
Host            = localhost
Database        =
ConfigFile      =
```

MySQL

```
[MySQL-test]
Description     = MySQL test database
Trace       = Off
TraceFile   = stderr
Driver      = MySQL
SERVER      = 192.168.1.26
```

```
USER        = pharvey
PASSWORD    =
PORT        = 3306
DATABASE    = test
```

NNTP driver

```
[nntp Data Source]
Description     = nntp Driver
Driver          = nntp Driver
Trace           = No
TraceFile       =
Host            = localhost
Database        =
Port            =
```

FreeTDS driver

```
Driver = TDS
Description = Northwind sample database
Trace = No
Server = 192.168.1.25
Database = Northwind
UID = sa
```

Sybase SQL Anywhere 5.0

Thanks Greg.

```
[Sybase SQL Anywhere 5.0]
Driver=Sybase SQL Anywhere 5.0
Description=Sybase SQL Anywhere 5.0 ODBC Driver
Userid=dba
Password=sql
DatabaseFile=sademo.db
```

Hopefully this will be of some use to someone... Nick Gorham