



This document is part of LIBPF

Paolo Greppi - 3iP © Copyright 2005 - 2008

All rights reserved; do not distribute without permission.

LIBPF

Installing on Linux

Summary

| | |
|--------------------------------|---|
| Additional packages..... | 2 |
| Configure the database..... | 3 |
| Configure ODBC..... | 5 |
| odbc.ini..... | 5 |
| odbcinst.ini..... | 6 |
| Set up LIBPF build tree..... | 6 |
| Get bimap..... | 7 |
| Get CSparse..... | 7 |
| Get silo..... | 8 |
| Get Gmm++..... | 8 |
| Get sundials..... | 8 |
| Get the LIBPF source code..... | 8 |
| Compile..... | 9 |
| Run..... | 9 |



This document describes the prerequisites and the steps for the installation on Linux of the LIBPF library and development environment, version 0.8.x.

The supported Linux distros are Debian Etch (4.0r3) and Lenny (5.0).

Additional packages

Install the following additional packages using apt-get or Synaptic:

- wget 1.10.2-2
- subversion 1.4.2dfsg1-2
- graphviz 2.8-2.4
- graphviz-dev 2.8-2.4
- g++ 4:4.1.1-15
- libloki-dev 0.1.5-2
- bjam 3.1.13-1
- boost-build 2.0-m11-2
- libboost-dev 1.33.1-10
- libboost-graph-dev 1.33.1-10
- postgresql-7-4 1:7.4.19-0etch1
- iodbc 3.52.4-5
- libiodbc2 3.52.4-5
- libiodbc2-dev 3.52.4-5
- odbc-postgresql 1:0.8.01.0200-2.1
- xsltproc 1.1.19-1

Currently these libraries are not available as sufficiently up to date Debian (deb) packages so we need to download and compile manually:

- libufsparse-dev
- libsundials-serial-dev

while no package at all exists for the silo (a data format and library developed at Lawrence Livermore National Laboratory for storing rectilinear, curvilinear, unstructured, or point meshes in 2D and 3D) and Gmm++ (a generic C++ template library for sparse, dense and skyline matrices).



Finally the bimap library is scheduled for inclusion in boost in the next release so we have to pull it in manually.

Configure the database

Log in as root and edit postgresql.conf to Allow TCP/IP Connections (this will be postgresql 7.4 on Etch):

```
vi /etc/postgresql/8.3/main/postgresql.conf
```

add the following line at the beginning of the Connection Settings section:

```
tcpip_socket = true
```

Next log in as postgres administrative user and create the database user libpf

```
su - postgres  
createdb retention
```

Now create the required tables:

```
psql retention  
DROP TABLE CATALOG;  
CREATE TABLE CATALOG (  
    GID          character (50),  
    ID           integer PRIMARY KEY,  
    TAG          character (50),  
    DESCRIPTION  character (255),  
    TYPE        character (50),  
    COMPLETETAG character (50),  
    PARENT      integer REFERENCES CATALOG (ID)  
);  
DROP TABLE ITBL;  
CREATE TABLE ITBL (  
    ID           integer PRIMARY KEY,  
    CATALOGID   integer REFERENCES CATALOG (ID),  
    TAG         character (50),  
    DESCRIPTION  character (255),  
    I           integer  
);
```



```
DROP TABLE ETBL;
CREATE TABLE ETBL (
    ID                integer PRIMARY KEY,
    CATALOGID        integer REFERENCES CATALOG (ID),
    TAG               character (50),
    DESCRIPTION       character (255),
    I                integer REFERENCES CATALOG (ID)
);
DROP TABLE QTBL;
CREATE TABLE QTBL (
    ID                integer PRIMARY KEY,
    CATALOGID        integer REFERENCES CATALOG (ID),
    TAG               character (50),
    DESCRIPTION       character (255),
    Q                double precision,
    UOM               character (50),
    INPUT             boolean,
    OUTPUT            boolean
);
DROP TABLE STBL;
CREATE TABLE STBL (
    ID                integer PRIMARY KEY,
    CATALOGID        integer REFERENCES CATALOG (ID),
    TAG               character (50),
    DESCRIPTION       character (255),
    S                character (255)
);
DROP TABLE TC;
CREATE TABLE TC (
    id                SEQUENCE PRIMARY KEY,
    start             integer REFERENCES CATALOG (ID),
    finish            integer REFERENCES CATALOG (ID),
    pathlength        integer
);
```



Finally give to the libpf user (used by the calculation kernel to access the database) and to any other user who might use LIBPF on the workstation (in that case, use a postgres sql user name that duplicates the local account login name and an empty password) the permissions to manipulate the retention tables, quit psql and exit the postgres session.

```
psql retention
create user libpf password 'test';
grant all on catalog to libpf;
grant all on itbl to libpf;
grant all on qtbl to libpf;
grant all on etbl to libpf;
grant all on stbl to libpf;
grant all on tc to libpf;
grant all on tc_id_seq to libpf;
\q
exit
```

Configure ODBC

Note: rather than modifying the odbc.ini and odbcinst.ini files you can also use the graphical tool iodbcadm-gtk.

odbc.ini

This is either in /etc or in /home/myuser/.odbc.ini

```
[ODBC Data Sources]
retention = PostgreSQL ODBC driver (Unicode version)
ret       = PostgreSQL ODBC driver (ANSI version)

[retention]
Driver    = /usr/lib/odbc/psqlodbcw.so
Database  = retention
ReadOnly  = No
Servername = localhost
Description = LIBPF Database
Port      = 5432
```



```
Protocol      = 6.4

[ODBC]
Trace         = 0
TraceAutoStop = 0
TraceFile     = /home/myuser/sql.log

[ret]
Driver        = /usr/lib/odbc/psqlodbc.so
Description   = LIBPF Database
Database      = retention
Port         = 5432
Servername    = localhost
```

odbcinst.ini

This is either in `/etc/odbcinst.ini` or in `/home/myuser/.odbcinst.ini`

```
[ODBC Drivers]
PostgreSQL ODBC driver (Unicode version) = Installed
PostgreSQL ODBC driver (ANSI version)    = Installed

[PostgreSQL ODBC driver (Unicode version)]
Driver  = /usr/lib/odbc/psqlodbcw.so
Setup   = /usr/lib/odbc/libodbcpsqlS.so
CommLog = 1
Debug   = 0

[PostgreSQL ODBC driver (ANSI version)]
Driver  = /usr/lib/odbc/psqlodbc.so
Setup   = /usr/lib/odbc/libodbcpsqlS.so
CommLog = 1
Debug   = 0
```

Set up LIBPF build tree

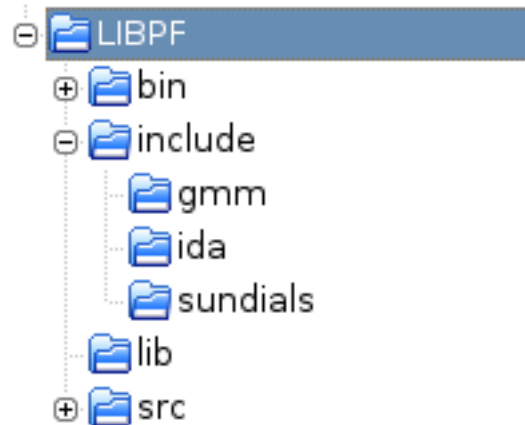
Create a LIBPF folder, then the following subfolders:

```
mkdir include
```



```
mkdir src
mkdir include
mkdir lib
mkdir bin
```

The LIBPF build tree at the end of the process (i.e. once the subsequent steps are completed) will look like this:



Get bimap

download bimap (an STL-like bidirectional map by Joaquín M López Muñoz) from:

<http://www.codeproject.com/KB/stl/bimap.aspx>

copy bimap.h in include/

Get CSparse

Get the CSparse library from the Tim Davis website:

<http://www.cise.ufl.edu/research/sparse/CSparse/>

and make it.

```
wget http://www.cise.ufl.edu/research/sparse/CSparse/versions/CSparse-2.2.1.tar.gz
tar zxf CSparse-2.2.1.tar.gz
cd CSparse
make
```

make install or manually copy the libcsparse.a file in lib/ and the cs.h file in include/.



Get silo

Get the silo archive from the Visit website:

<https://wci.llnl.gov/codes/visit/>

unpack, configure and make:

```
wget https://wci.llnl.gov/codes/visit/3rd_party/silo060605.sh
sh silo060605.sh
cd silo060605
./configure
make
```

manually copy the libsilo.a file in lib/ and the silo.h file in include/.

Get Gmm++

```
wget http://download.gna.org/getfem/stable/gmm-3.0.tar.gz
```

Extract and copy the contents of the include folder (a gmm subfolder) in include/.

Get sundials

Get the sundials-2.3.0.tar.gz file from the Sundials home:

<https://computation.llnl.gov/casc/sundials/main.html>

Extract, configure, make then manually copy the ida and sundials folders in include/.

Also copy sundials_config.h in include/sundials.

Finally copy the file libsundials_ida.a in lib/.

Get the LIBPF source code

Simply check out the LIBPF source tree using the subversion client:

```
svn checkout https://xxxxx/LIBPF/ src
```

You will be asked to accept the server SSL certificate (it is suggested to accept permanently the certificate) and to enter the username / password you have received.

We need to generate the header and code for smartenums classes using the XSLT command line processor:

```
xsltproc smartenum_hpp.xslt smartenums.xml > smartenum.hpp
```



```
xsltproc smartenum_cc.xslt smartenums.xml > smartenum.cc
```

Compile

The compilation of LIBPF is automated using the bjam (a make replacement that makes building simple things simple and building complicated things manageable, from the boost community).

The bjam tool has to be set up once for all to find your compiler:

```
vi /home/myuser/user-config.jam
```

add the line:

```
using gcc : 4.1 : gcc ;
```

You can inspect the src/Jamfile to know what targets you can build. The typical build commands are:

```
cd src  
bjam Qsale_pepe  
bjam Qsale_pepe release
```

Run

Bjam will place the compiled executables in the bin subfolder corresponding to your compiler / options combination.

It is advisable to run in the bin folder, to avoid clogging the src folder with the temporary files.

A typical command is:

```
cd ../bin  
gcc-4.1/debug/Qsale_pepe
```