OpenDSS is a distribution system simulator that is generally comparable to GridLAB-D, although each program has advantages and disadvantages over the other [1]. It was developed with EPRI funding as a Windows-only application, using commercial Windows-only tools. This memo summarizes the build process of a console-mode, cross-platform version using GCC, FPC and Lazarus. Tested platforms include Windows 10, Ubuntu 16.04 and Mac OS X High Sierra.

OpenDSS was written in Delphi (a Windows-specific version of Pascal), and it requires the KLUSolve sparse matrix solver, which was written in C/C++. After compiling KLUSolve with GCC, it’s then possible to build OpenDSS with Free Pascal (FPC) 3.0 on Windows, Linux and Mac OS X. Modern versions of Pascal are object-oriented, but simpler than C++ and Java. There are syntax differences, but a developer who is already comfortable with C++ or Java can be productive in Pascal, too. See <http://wiki.freepascal.org/Lazarus_Documentation#Lazarus_and_Pascal_Tutorials>. There is a Free Pascal IDE and cross-platform GUI toolkit called Lazarus, but this is not required for OpenDSS.

With a common code base, there can still be differences between the native and cross-platform versions. For example, EPRI has implemented a Windows-only parallelization scheme, and continues to develop Windows-only automation interfaces. NREL uses a shared-library version of OpenDSS called by Python and/or Julia. On the other hand, PNNL is implementing a FNCS interface and command-line history only for the cross-platform version. However, both versions maintain the same core modeling and analysis functions.

[1] R. C. Dugan and T. E. McDermott, "An open source platform for collaborating on smart grid research," in *Power and Energy Society General Meeting, 2011 IEEE*, 2011, pp. 1-7.

## Running Cross-platform OpenDSS

1. Enter “opendsscmd” from a command prompt
   1. The program’s >> prompt will appear. Enter any OpenDSS command(s) from this prompt
   2. Up and down arrows navigate through the command history
   3. Enter “help” from the >> prompt for the built-in help
   4. Enter “exit”, “q” or an empty line from >> to exit
2. You can enter “opendsscmd filename.dss” from a command prompt. This runs the OpenDSS commands in filename.dss, and then exits immediately.
3. You can enter “opendsscmd –f” from a command prompt; this enters a FNCS time step loop.
4. You can enter “opendsscmd –f filename.dss” from a command prompt. This runs the OpenDSS commands in filename.dss, and then enters a FNCS time step loop.

## Open Issues

1. The regular expressions for the batchedit command, which are implemented in ExecHelper.pas, have become case-sensitive. They need to be made case-insensitive.

## Code Repositories

The source code is under version control at SourceForge and GitHub. The code repositories for OpenDSS, the sparse matrix solver and the command-line history are at:

* <https://github.com/pnnl/linenoise-ng.git>
* <https://sourceforge.net/projects/klusolve/>
* <https://sourceforge.net/projects/electricdss/>

It’s assumed that all three repositories will be cloned under a common location, such as ***~/src***

## Streamlined Build Instructions

These instructions assume that GridLAB-D and (optionally) FNCS already build on the target computer. CMAKE and an SVN client are also required. These instructions were tested on Mac OS X High Sierra, Ubuntu 16.04 and Windows 10. On Windows, if not building GridLAB-D, the required preliminary steps are described in a later section. On Windows, some steps must be done from an MSYS terminal, not the regular Command Window. The Lazarus IDE may be of interest for those actively developing for OpenDSS, and this is also described later.

1. Install FPC 3.0 or higher:
   1. Linux: ***sudo apt-get install fpc***
   2. Windows: from <https://sourceforge.net/projects/freepascal/files/Win32/3.0.4>, you need the base win32 installation and then the x86\_64 cross-compiler
   3. Mac: from [https://sourceforge.net/projects/freepascal/files/Mac OS X/3.0.4/](https://sourceforge.net/projects/freepascal/files/Mac%20OS%20X/3.0.4/), you need the intel-macosx.dmg installer
2. Build linenoise-ng (starting in ~/src or c:) (use MSYS on Windows)

***mkdir linenoise-ng***

***cd linenoise-ng***

***git clone https://github.com/pnnl/linenoise-ng.git .***

***mkdir build***

***cd build***

***cmake -DCMAKE\_BUILD\_TYPE=Release ..***

*on Windows, add* ***-G”MSYS Makefiles”***

***make***

***sudo make install***

* 1. Linux only (may defer this to step 3f below): ***ldconfig***

1. Build KLUSolve (use MSYS on Windows)
   1. ***mkdir KLUSolve***
   2. ***cd KLUSolve***
   3. ***svn checkout https://svn.code.sf.net/p/klusolve/code/ .***
   4. Manually ***mkdir Lib*** or ***md Lib***
   5. Mac OS X only: manually ***mkdir KLUSolve/Obj***
   6. Issue ***make all*** (or ***mingw32-make all*** on Windows without a GridLAB-D build)
      1. On Mac OS X ***make all*** usually fails to link the first time due to a race condition on copying files. If that happens, just ***make all*** again.
   7. Manually install the shared object library (TODO: put this in the Makefile)
      1. Linux static: ***cp Lib/libklusolve.so /usr/local/lib*** and then ***ldconfig***
      2. Windows: copy Lib/libklusolve.dll to the target location, either the opendsscmd location or a Windows system directory
      3. Mac: ***cp Lib/libklusolve.dylib /usr/local/lib***
   8. Test the demo program:
      1. Linux, Mac OS X, or MSys prompt: from Test subdirectory ***bash run\_concat.sh***
         1. If the unzip command is not available on Windows MSYS, try step ii below
      2. Windows Command Prompt: from Test subdirectory, manually unzip kundert\_test\_matrices.zip and then invoke ***run\_concat.bat***
2. Linux: symbolic links are probably required, based on the contents of /usr/lib. For example:
   1. sudo ln -sv /usr/lib/x86\_64-linux-gnu/libstdc++.so.6.0.21 /usr/lib/x86\_64-linux-gnu/libstdc++.so
   2. sudo ln -sv /lib/x86\_64-linux-gnu/libgcc\_s.so.1 /lib/x86\_64-linux-gnu/libgcc\_s.so
3. Extract the selected OpenDSS source code (starting in ~/src or c:). Because EPRI keeps large build products, Windows-only artifacts and copy-paste code branches in the same repository, two different strategies are provided. (Note: on Windows, this step can be done in either MSYS or the Command Prompt).
   1. ***mkdir OpenDSS*** or ***md OpenDSS***
   2. ***cd OpenDSS***
   3. Strategy 1: grab everything with
      1. svn checkout https://svn.code.sf.net/p/electricdss/code/trunk .
   4. Strategy 2: selective retrieval (could be the basis of a cross-platform installer)
      1. svn checkout --depth immediates https://svn.code.sf.net/p/electricdss/code/trunk .
      2. svn update --set-depth infinity Doc
      3. svn update --set-depth infinity Test
      4. svn update --set-depth infinity Distrib/Doc
      5. svn update --set-depth infinity Distrib/EPRITestCircuits
      6. svn update --set-depth infinity Distrib/IEEETestCases
      7. svn update --set-depth infinity Distrib/Examples
      8. svn update --set-depth infinity Source/Common
      9. svn update --set-depth infinity Source/CMD
      10. svn update --set-depth infinity Source/DDLL
      11. svn update --set-depth infinity Source/Controls
      12. svn update --set-depth infinity Source/Executive
      13. svn update --set-depth infinity Source/General
      14. svn update --set-depth infinity Source/Meters
      15. svn update --set-depth infinity Source/Shared
      16. svn update --set-depth infinity Source/Parser
      17. svn update --set-depth infinity Source/PCElements
      18. svn update --set-depth infinity Source/PDElements
4. Build and test OpenDSS
   1. **cd *./Source/CMD*** in your command prompt
   2. ***mkdir units*** or ***md units***
   3. ***./build.sh*** or ***build.bat***
   4. ***cd test***
   5. ***./opendsscmd*** and see if the command history works. Type “q” or ctrl-C to exit.
      1. libgcc\_s\_seh-1.dll might not be found on Windows. In that case, unzip <https://github.com/pnnl/tesp/blob/master/install/Windows/MinGWredist.zip> into c:\opendsscmd
   6. ***./opendsscmd IEEE13Nodeckt.dss***. It should solve a 13-bus circuit, exit, and open a text editor on the voltage results. However, this command may fail to find the circuit, especially if you skipped step 6e. In that case, try again. This is Open Issue #3.
   7. If FNCS is installed, test that connection with ***./test\_fncs.sh*** or ***test\_fncs.bat***. Look for results in ***\*.log*** and ***tracer.out***.

## Using Lazarus instead of FPC from the Command Line

Instead of the build scripts, you will open the project file ~/src/OpenDSS/Source/CMD/opendsscmd.lpi from the Lazarus IDE. The IDE provides more convenient management of project files, builds and error messages. To install the IDE on Ubuntu, ***sudo apt-get install lazarus*** works. On Windows and Mac, you can download a combined package of Lazarus 1.6 and FPC 3.0 from <http://www.lazarus-ide.org/>

On the Mac, pay close attention to <http://wiki.freepascal.org/Installing_Lazarus_on_MacOS_X> for setting up gdb. When you start the Lazarus IDE for the first time; it should find the debugger (gdb) and possibly two compilers. **Choose the fpc compiler**, not the default ppc386 compiler. Otherwise, you can only make 32-bit executables from Lazarus. If necessary, you can fix this later from the IDE Tools / Options menu. However, the Lazarus IDE on Mac does not fully support Carbon, meaning that it can practically only create 32-bit GUI applications. This is a significant barrier to the possible cross-platform GUI for OpenDSS, at least one based on free development tools.

## Preliminary Steps for Windows; not building GridLAB-D

These were tested on Windows 10 and Windows 7:

1. Download and install 64-bit GCC from <http://tdm-gcc.tdragon.net>
2. Download and install just MSYS 1.0.11 from [www.mingw.org/wiki/msys](http://www.mingw.org/wiki/msys). During the “post install” procedure:
   1. answer [y] that MinGW is installed
   2. answer c:/TDM-GCC-64 for the location of MinGW
3. If you don’t have an SVN client, install from <https://tortoisesvn.net/>
   1. On the “Custom Setup” page, select “Command Line Tools” for inclusion
4. If you don’t have a Git client, install from <https://www.sourcetreeapp.com/>
   1. To use the command line Git from SourceTree on Windows, you may use the “Actions/Open in Terminal” menu command
   2. You may also install the command line Git from <https://git-scm.com/downloads>
5. If you don’t have Cmake, install from <https://cmake.org/>
   1. Choose to add Cmake to the system path
   2. Change location to c:\cmake\, which is better for MSYS

## Other OpenDSS Development Branches

EPRI’s parallelization, which is Windows only and requires Delphi.

* <https://sourceforge.net/p/electricdss/code/HEAD/tree/trunk/Parallel_Version/>

A mirrored-repository Makefile:

* [https://github.com/Muxelmann/OpenDSSDirect.make](https://urldefense.proofpoint.com/v2/url?u=https-3A__github.com_Muxelmann_OpenDSSDirect.make&d=DwMGaQ&c=YFYuafCCopBdR2aI1UDiwKbQTSrP7gdpddSkt1TYoDc&r=dL7HKtkmKadd2sirF916fQ&m=qQcQH5QMEwNBB85vO8Tp_cLUfRZGWnTjW0jY4Hu78ik&s=eYRDXbob3jZKN9FicljUs6TT5Ig5Xvu7GDumgG2exGY&e=)

NREL’s Python interface to the direct-call, shared library version:

* [https://github.com/NREL/OpenDSSDirect.py/](https://urldefense.proofpoint.com/v2/url?u=https-3A__github.com_NREL_OpenDSSDirect.py_&d=DwMGaQ&c=YFYuafCCopBdR2aI1UDiwKbQTSrP7gdpddSkt1TYoDc&r=dL7HKtkmKadd2sirF916fQ&m=qQcQH5QMEwNBB85vO8Tp_cLUfRZGWnTjW0jY4Hu78ik&s=aJIvErKkbESgWPX9k0_AqGjYibeOgB48E7zVlb4Pa2A&e=)

EPRI’s Julia interface to the direct-call, shared library version:

* [https://github.com/tshort/OpenDSSDirect.jl](https://urldefense.proofpoint.com/v2/url?u=https-3A__github.com_tshort_OpenDSSDirect.jl&d=DwMGaQ&c=YFYuafCCopBdR2aI1UDiwKbQTSrP7gdpddSkt1TYoDc&r=dL7HKtkmKadd2sirF916fQ&m=qQcQH5QMEwNBB85vO8Tp_cLUfRZGWnTjW0jY4Hu78ik&s=Bk8MtimWLCZ-dny4Gxq8kLPLjDPQzU12Ud2h3vVDARA&e=)

These last two NREL and EPRI interfaces to the shared library are using mirrored repositories. If these interfaces are of interest to PNNL developers, please use one of the following options, both of which reference the main repository:

* From ~/src/OpenDSS/Source/DDLL, use ***./build.sh*** or ***build.bat***
* From the Lazarus IDE, open the project file ~/src/OpenDSS/Source/DDLL/OpenDSSDirect.lpi