# $Id: README,v 1.3 2017/06/19 20:00:10 pm Exp $

Basic Seismic Utlilities (BSU)
Author: Dr. P. Michaels, PE <pm@cgiss.boisestate.edu>
Center for Geophysical Investigation of the Shallow Subsurface
Boise State University
Boise, Idaho 83725
VERSION: 3.0.0
FORMATS: DEBIAN and UBUNTU (*.deb), REDHAT and CENTOS (*.rpm)
         and TARBALL (autoconf, automake, libtool)
WHICH FORMAT TO USE:
   A). If you plan on modifying or extending the software:
       Use the source TAR archive, install under /usr/local.
   B). If you just want the binaries, use a package and
       install under /usr (Debian policy, also good for RPM)

ABSTRACT:
This package is a collection of seismic signal processing and seismic modeling software. It is designed for problems in Engineering Geophysics. The binary data format is derived from SEG-Y (no reel header, 240 byte trace header, data). It is very compatible with Seismic Unix, but there are some differences in the headers (3-component recording, polarized sources, etc.).

---

CONTENTS:

---

## 1. QUICK START

1.0 Install any BSU dependencies. These would be libraries that BSU uses (see section 2 below). On Unix/Linux systems, this includes BLAS, LAPACK, GSL, PLPLOT, GNUPLOT (and packages that these may depend on.

### 1.1 BINARY ONLY INSTALL

```
    1.1.0 Mingw32 MS Windows
       Precompiled binaries
          Unzip: BSU_EXEC.zip in a location suitable for executable
          programs.
          example: C:\Programs
                     unzip BSU_EXEC.zip
        NOTES: 1). This will install binaries under C:\Programs
               2). Optional: install GNUPLOT from
               http://www.gnuplot.info/index.html
               3). Optional: MingW32 to add unix commands to Powershell
               http://www.mingw.org/
               4). Optional: Octave
               https://wiki.octave.org/Octave_for_Microsoft_Windows


    1.1.1 From a PACKAGE (* replaced with your specific downloaded
    package):
        Debian: sudo dpkg -i bsu-3.0.0*.deb
        RPM: sudo rpm -ihv bsu-3.0.0*.rpm
        NOTES: 1). RPM is relocateable, can use prefix option to change
        install
               directory tree.  Example: sudo rpm -ihv --prefix /usr/local
               bsu-3.0.0*.rpm
               2). DEB package is policy compliant, installs under
               /usr tree.
               3). Default for RPM is install under /usr tree.

 1.2 SOURCE, COMPILE and INSTALL
    1.2.0 From a TARBALL (Debian, Mint, Ubuntu, Arch, Slackware, Redhat)
                cd /usr/local/src
                tar xvzf bsu-3.0.0.tar.gz
                cd bsu-3.0.0
                ./configure <options>
                make install
        NOTES: 1). To view options, ./configure -h | less
               Make your choice (see bsu-user-guide3.pdf),
               Then "make" and "make install".
               2). To remove, "make uninstall".  To recompile, "make
               clean" followed
               by "./configure <options>", followd by "make" and
               "make install".

WHAT IF I CAN'T WRITE IN /usr/local?
If you don't have write privileges in /usr/local, you
can build it in your home directory.  In that case you will want to use
the "prefix"
options when you run ./configure from inside the bsu-3.0.0 directory.
Example: (here, "mydirectory" would be your home directory name)
         mkdir -p /home/mydirectory/local/bin
         mkdir -p /home/mydirectory/local/share
         mkdir -p /home/mydirectory/local/lib
         mkdir -p /home/mydirectory/local/include
         cd bsu-3.0.0
         ./configure --prefix=/home/mydirectory/local <other options>
         make
         make install
```

---

## 2. DEPENDENCIES

These are software libraries that BSU takes advantage of. Not every BSU program needs these, but a complete installation requires these packages.

### 1.0 BLAS and LAPACK

http://www.netlib.org/blas/
http://www.netlib.org/lapack/

Debian, Redhat, and most other modern Linux distributions have these packages available. You will not usually need to compile these. In most cases, you just need to install LAPACK, since the package system will automatically add BLAS (which is required by LAPACK). For example,

DEBIAN & UBUNTU:

```
  sudo apt-search lapack
```

will give you a list of lapack packages. One can also run aptitude or synaptic to find these packages. With synaptic (GUI), use the "search" button.

REDHAT & CENTOS:

You can use the yum tool, or perhaps yumex (GUI tool which is like synaptic in debian).

Look for the "lapack3" version, and if available, any development packages if you plan on compiling BSU from source (these often end with names *-dev or *-devel).

CAUTION: Fortran is moving from the g77 to the gfortran compiler. Make sure that if you plan on using a compiled LAPACK library, it has been compiled with the fortran compiler you plan on using for BSU. Package names that end with *-gf in Debian or Ubuntu systems have been compiled with gfortran. Debian 5.0 (Lenny) only comes with gfortran. Debian 4.0 (Etch) with both g77 and gfortran, and will tend to use g77.

EXAMPLES:

On a Debian 9 "stretch" System, packages installed were:

```
liblapack-dev
liblapack3
libblas-common
libblas-dev
libblas3
libopenblas-base
libgsl-dev
libgsl2:amd64
libshp-dev:amd64
libshp2:amd64
libplplot-c++11:amd64
```

```
libplplot-dev:amd64
libplplot-fortran10:amd64
libplplot-lua:amd64
libplplot-ocaml
libplplot12:amd64
plplot-doc
plplot-tcl:amd64
plplot-tcl-bin
plplot-tcl-dev:amd64
plplot12-driver-qt:amd64
plplot12-driver-wxwidgets:amd64
plplot12-driver-xwin:amd64
python-plplot
python-plplot-qt
liboctave-dev
liboctave3v5:amd64
octave
octave-common
octave-control
octave-info
octave-nan
octave-signal
octave-tsa

NOTE: Not all these plplot packages are needed for BSU, but this is what
I had on my system. The *doc and *test packages are not required, but may
be useful.
------------------------------------------------------------------------
For Cross-Compile on Debian 9 Host: Target Microsoft Windows 8.1 64 bit
Packages:
binutils-mingw-w64
binutils-mingw-w64-i686
binutils-mingw-w64-x86-64
g++-mingw-w64
g++-mingw-w64-i686
g++-mingw-w64-x86-64
gcc-mingw-w64
gcc-mingw-w64-base
gcc-mingw-w64-i686
gcc-mingw-w64-x86-64
gdb-mingw-w64
gdb-mingw-w64-target
gfortran-mingw-w64
gfortran-mingw-w64-i686
gfortran-mingw-w64-x86-64
gnat-mingw-w64
gnat-mingw-w64-base
gnat-mingw-w64-i686
gnat-mingw-w64-x86-64
mingw-w64-common
mingw-w64-i686-dev
mingw-w64-tools
mingw-w64-x86-64-dev

-------------------------------------------------------------------------------
FOR RPM BASED SYSTEMS
```

The general rule is that a packagename-dev development package will have a name like packagename-devel in RPM based systems (difference is -devel instead of -dev).
Some Examples:

```
plplot-5.9.7-3.el6.1.x86_64
octave-3.4.3-2.el6.x86_64
gsl-1.13-1.el6.x86_64
gsl-devel-1.13-1.el6.x86_64
lapack-3.2.1-5.el6.x86\_64
lapack-devel-3.2.1-5.el6.x86_64
blas-3.2.1-5.el6.x86_64
blas-devel-3.2.1-5.el6.x86_64
gnuplot-4.2.6-2.el6.x86_64
```

---

## BUILDING SOURCE PACKAGES

### 3. LINUX SYSTEMS (Source Package Compile)

These are instructions on rebuilding from a source package. One can download either a *.deb or *.rpm source package and build it with these instructions. For MS-Windows, I built BSU with the MingW32 tool chain (see above package list) on a Linux host.

### 3.0 From a DEBIAN PACKAGE

Before building a source package, you should have these packages on your system:

```
debian-policy
debhelper
dh-make
devscripts
lintian
```

Debian: There are 3 or more files in a typical Debian package (in directory where you have bsu-3.0.0*.dsc, *.tar and any *.diff files)
Run the command which extracts a source directory tree:

```
dpkg-source -x bsu-3.0.0.dsc
```

Then cd into bsu-3.0.0 created by above command. Then cd into debian directory, edit changelog file if needed. You may also wish to edit the control, rules, and bsu.manpages files. Then exit the debian directory (return to next level above).

Run the command:

```
dpkg-buildpackage -rfakeroot -uc -us
```

This will build a new bsu*.deb package in your directory one level above.

Install that with the command (the * represents your version number)

```
sudo dpkg -i bsu*.deb
      OR
sudo gdebi bsu*.deb
```

3.1 From a RPM PACKAGE

RedHat packages are a single file, typically ending in *.src.rpm and can be built either in /usr/src/redhat directory, or in your own home directory. For example, to build a package in my home directory, I have a .rpmmacros file in ${HOME}. Files which begin with a dot "." are hidden unless you use the -a option on an ls command.

EXAMPLE .rpmmacros file

```
pm:~$ cat .rpmmacros

%packager P. Michaels <pm@cgiss.boisestate.edu>
%vendor BSU
%_topdir /home/pm/redhat
%_prefix /usr
%_exec_prefix %{_prefix}
%_mandir %{_prefix}/share/man
%_datadir %{_prefix}/share
%_sysconfdir %{_prefix}/etc
%_bindir %{_exec_prefix}/bin
```

In the above example, the source package will be installed in /home/pm/redhat (handy if you don't have write privileges in /usr/src/redhat). The result of compilation will be a binary package to be installed under /usr (set by the %_prefix). A typical directory tree looks like this:

EXAMPLE Say we want to build the GSL source package on a Ubuntu AMD64 machine. The directory structure would look like this:

```
pm:~/redhat$ tree
.
|-- BUILD
|-- RPMS
|  |-- i386
|  |-- noarch
|  `-- x86_64
|-- SOURCES
|  |-- gsl-1.1-nousr.patch
|  |-- gsl-1.10-head.patch
|  |-- gsl-1.10-lib64.patch
|  |-- gsl-1.10.tar.gz
`-- SPECS
    `-- gsl.spec
```

This directory tree was created with the command

```
rpm -ihv gsl*src.rpm
```

because my .rpmmacros file specified the location under my home directory.
This step is comparable to the "dpkg-source -x" command in Debian.

The next step is to cd ~/redhat/SPECS directory, and then issue a build command:

```
rpmbuild -bb gsl.spec
```

(for example would build only a binary package, -bb option ).
RPM building is a big topic, see "Maximum RPM" by Edward C. Bailey for more details.
www.redhat.com/docs/books/max-rpm/index.html

## 4. COPYRIGHT Information, GPL License

COPYRIGHT: