

Install Seismic Unix on Ubuntu to /usr/su44r19

First steps

Introduction

This document is about installing Seismic Unix (SU) to the system-level directory

`/usr/su44r19`

This is more complicated than installing SU to a user directory; for example,

`/home/david`

A reason to install Seismic Unix (SU) to a personal user area is because, usually, system permission is not needed to install software there. A reason to install SU to a “higher level” location is because the SU software is not in the personal “/home/david” space. I wrote another document about installing SU to a personal directory.

Get the Seismic Unix tar file

To get the latest Seismic Unix tar (compressed) file, search “seismic unix wiki”

<https://wiki.seismic-unix.org/start>

Click “Download Seismic Un*x”

Click the Download button with the file name: `cwp_su_44R19.tgz`

This tar file will “probably” go into my `~/Downloads` directory

Note that the tilde (~) refers to my user directory; for example:

`/home/david`

is the same as

`~`

So, this person’s Downloads directory is

`/home/david/Downloads`

or

`~/Downloads`

Browser: Open an SU installation page

Go to the Seismic Unix WIKI site:

<https://wiki.seismic-unix.org/start>

Click “Documentation”

Click “SU installation”

The installation instructions in this document and in the Seismic Rocks YouTube video are a hybrid of this web page and an installation web page that used to be available on the SU GitHub “Wiki” site:

<https://github.com/JohnWStockwellJr/SeisUnix/wiki>

Optional section: .bash_aliases, a hidden file

A hidden file is any file that begins with a period. The same is true of a hidden directory. Files and directories are hidden because system people think you and I should not change them or even know they exist. Most of the time, this is reasonable thinking. However, later in this document, I will show you how I change the bash resource (.bashrc) file as part of the SU installation. And, this section show how I create and use a hidden file that will have useful aliases. Aliases are keyboard shortcuts; that is, a short command that replaces a long one, and is readily available because it is in the .bash_aliases file.

Below, do the command that follows the dollar sign (\$), the terminal prompt, to understand my suggestions.

```
$ ls -a                <-- view hidden files
$ cat .bashrc
```

The image below is part of the .bashrc file (line numbers added). The image shows that the .bashrc file runs the .bash_aliases file when I “source” the .bashrc file (lines 104-106). The “source” command is illustrated below.

```

99 # Alias definitions.
100 # You may want to put all your additions into a separate file like
101 # ~/.bash_aliases, instead of adding them here directly.
102 # See /usr/share/doc/bash-doc/examples in the bash-doc package.
103
104 if [ -f ~/.bash_aliases ]; then
105     . ~/.bash_aliases
106 fi
```

I am always very, very careful when I edit .bashrc because if I do something wrong here, it can be difficult to log in next time. ***Ouch!***

The next two sections are not part of the SU installation, but I recommend them to you.

Discuss an alias for “rm”

Before I start the installation, I *strongly* recommend (you do not have to do this) that you make an alias for the remove (“rm”) command. The default “rm” command does not let me change my mind. The default does not “confirm delete,” it only deletes. In the past, sometimes this has made me very sad. The following two examples demonstrate the value of changing “rm” from the default.

```
$ touch asdf.txt
```

```

$ ls -l asd*           <-- note the wildcard. Empty file
$ rm asdf.txt         <-- the file goes away! No "confirm"

$ man rm              <-- read the "-i" flag documentation
$ alias rm='rm -i'   <-- temporary change to "rm"
                       No spaces around the equal sign!

$ touch asdfg.txt
$ ls -l asd*         <-- empty file
$ rm asdfg.txt       <-- confirm!

```

The set of instructions above demonstrate the value of replacing the default “rm” command with an “rm” alias that confirms *remove*.

Put useful aliases in `.bash_aliases`

When creating an alias, do NOT use spaces around the equal (“=”) sign!

```

$ ls -a                <-- view hidden files
$ touch .bash_aliases <-- create the hidden file
$ gedit .bash_aliases <-- gedit is a good, simple editor

```

```
alias rm='rm -i'      <-- add this line to the file
```

Save, Exit

```
$ alias rm
```

```
<blank>
```

My system does not know about the alias for “rm” because I did not “source” my `.bashrc` file.

```
$ source .bashrc
```

```
$ alias rm
```

```
alias rm='rm -i'
```

I have other aliases, mainly for personal variations of “ls”. You can learn how these work by looking at the “man” page for “ls”.

```
alias ll='ls -lF'
alias lla='ls -alF'
alias llrt='ls -lFrt'
alias la='ls -a'
alias l='ls -CF'

```

Generally, I do not overwrite existing commands.

"rm" is an exception because I definitely *do* want to change the default "rm"

When I log in, .bashrc is automatically read, so all aliases are automatically up to date. But, if I add an alias using a terminal window, all other **open** terminal windows are NOT up to date.

After I change .bash_aliases, I use

```
$ source ~/.bashrc
```

in *all* my open terminal windows so they all can use the new aliases.

“source” the .bashrc file to use changes to hidden file “.bashrc”

“source” the .bashrc file to use changes to hidden file “.bash_aliases”

Return to installation discussion

Create the SU directory and untar the Seismic Unix software

The command below puts me at the “root” directory.

```
$ cd /
```

The screen capture below shows the result of my “ls -CF” command here (using an alias).

```
df@sp3:/$ l
bin@    dev/    lib@    libx32@  mnt/    root/    snap/    sys/    var/
boot/   etc/    lib32@  lost+found/  opt/    run/    srv/    tmp/
cdrom/  home/   lib64@  media/   proc/   sbin@   swapfile  usr/
df@sp3:/$
```

I “cd” into /usr. When I try to make an SU directory under /usr, I get an error:

```
df@sp3:/usr$ mkdir su44r19
mkdir: cannot create directory 'su44r19': Permission denied
```

Below, I used a different “ls” command (ls -lF) to see who owns these files and directories.

```
drwxrwxrwt  21 root root          4096 Jun 12 15:52 tmp/
drwxr-xr-x  14 root root          4096 Jun 11 16:00 usr/
drwxr-xr-x  14 root root          4096 Apr 23 01:42 var/
```

The answer is “root”. I am not “root”, I am user “df”. Also, I am not in root’s *group*.

Observe that directory /usr has permissions for user, group, others:

```
rwxr-xr-x
```

This means

- user root can read, write, execute (7)
- members of root’s group can read, execute (5)
- others can read, execute (5)

The numbers in the above three lines refer to the numeric way to represent these permissions. One reference for these permissions is Guru99:

<https://www.guru99.com/file-permissions.html>

To enable any user to create files and directories under `/usr`, I do this:

```
$ cd /
```

```
$ sudo chmod 777 usr
```

Now the permissions for directory `/usr` are:

```
drwxrwxrwt 21 root root 4096 Jun 12 15:47 cwp/
drwxrwxrwx 14 root root 4096 Jun 11 16:00 usr/
drwxr-xr-x 14 root root 4096 Apr 23 01:42 var/
```

- `u`ser root can read, write, execute (7)
- members of root's group can read, write, execute (7)
- `o`thers can read, write, execute (7)

Now I can continue with the installation. I make a directory to hold the SU software, copy the SU tar file to the new directory, and untar the file:

```
$ cd /usr
```

```
$ mkdir su44r19
```

```
$ cp ~/Downloads/cwp_su_all_44R19.tgz su44r19/.
```

```
$ cd su44r19
```

```
$ tar -zxvf cwp_su_all_44R19.tgz
```

Explore the SU subdirectories. Note that these directories cannot yet be used by my operating system. For example, if I enter

```
$ suplane
```

I get “command not found”.

Make a backup copy of my bash shell resource file `.bashrc`

Hidden file “`.bashrc`” is critical to my user space. So, before I edit it, I will make a backup copy.

```
$ cd ~
```

```
$ cp .bashrc .bashrc_backup
```

Add two variables to my `.bashrc` file

```
$ gedit .bashrc
```

Add the following two lines to `.bashrc`:

```
export CWPROOT=/usr/su44r19
```

```
export PATH=$PATH:$CWPROOT/bin
```

Save, Exit

The image below shows the last five lines of my `.bashrc` file (line numbers added).

```
118
119 # Seismic Unix
120 export CWPROOT=/usr/su44r19
121 export PATH=$PATH:$CWPROOT/bin
122
```

The first command is like setting an alias. The second command concatenates the SU library of executable files, the “bin” folder, into the entire path of user commands.

```
$ echo $CWPROOT
    <blank>
$ source .bashrc
$ echo $CWPROOT
    /usr/su44r19
```

`CWPROOT` is now a variable in my `.bashrc` file that will be used by the SU installation.

Make a backup copy of Makefile.config

File “`Makefile.config`” is critical to installing SU. But before I use it, I will make a backup copy because I am going to make one edit to it.

```
$ cd /usr/su44r19/src
$ cp Makefile.config Makefile.config.backup
```

Make one edit to Makefile.config

Open “`Makefile.config`” to edit the `XDRFLAG` option.

```
$ gedit Makefile.config
```

Around line 35 there is:

```
XDRFLAG = -DSUXDR
```

According to the “CWP feature options” described on lines 24 and 25, this setting “forces all SU data to be big endian independent of processor architecture.” But, by its nature, my Intel machine is little endian. I want any data I create in SU to be in the native “little endian” format, not just because my machine is little endian, but because, today, Intel (little endian) machines are very popular.

I will comment out this line, and replace it with a single blank space on the right side of the equal sign:

```
#XDRFLAG = -DSUXDR
XDRFLAG =
```

To repeat, put a single blank space on the right side of the equal sign.

Save, Exit

A reference for “endianness” is the Wikipedia entry:

<https://en.wikipedia.org/wiki/Endianness>

Future versions of Seismic Unix

Suppose there is a later version of SU, say, 45R2. What would I do with my current installation and how would I install the new version?

I would leave the current version alone, do nothing with it. Or, I might remove all of its directories, as I show in the next two lines:

```
$ cd /usr
$ rm -rf su44r19
```

Whether I remove the current version is not important (unless I am starving for space on my hard drive). The important tasks are (1) change the first “import” line in my .bashrc file to point to the new version directory and (2) source the .bashrc file so \$CWPROOT is redefined. After that, untar the new SU in its own directory, then follow the installation instructions below.

Finally, install SU with “make”

Change my location (“cd”) to the SU “src” directory

```
$ cd $CWPROOT/src
```

Realize this is the same as:

```
$ cd /usr/su44r19/src
```

I will execute “make” several times. The list below is in the comments of the “Makefile” file (not the “Makefile.config” file).

```
$ make install          main SU code
$ make xtinstall       X-toolkit code
$ make utils           utilities code
$ make finstall        FORTRAN code
$ make mglinstall      Mesa/OpenGL code
$ make xminstall       extra (optional) X-Motif code
$ make sfinstall       SFIO materials and “segdread”
```

Something you don’t need to know ... In the “src” directory, executing any of the “make” instructions; for example,

```
$ make install
```

runs “Makefile” which uses “Makefile.config”. Depending on which of the “make” I execute, a different part of “Makefile” is used.

Only the success of the first two “make” are required for a successful installation of SU. All other “make” are optional. I suggest you try them all (in order, one at a time) for the sake of as much SU installation as possible, but know that only the first two “make” are necessary for a good time with SU.

In the “make” sections below, first is the command, then suggestions for software installation when there is an error message.

- Look carefully for error messages and the system messages for needed software.
- A software installation command can be entered more than once. Ubuntu will not re-install a program; that is, there is no harm trying the same install command again.
- During a successful install, there are a lot of system *warnings*, not errors. Ignore warnings.

1. make install main SU code

```
$ make install
```

Command “make” not found? It can be installed with either of these commands:

```
$ sudo apt install make
```

```
$ sudo apt install make-guile
```

```
$ make install
```

Continue install? [y]

Hit Enter, then hit the space bar to move quickly through the installation notice.

Agree to abide by the terms of the LEGAL STATEMENT? [y]

Send automatic e-mail to John? [y]

Do you have an install error with gcc compiler?

```
$ sudo apt install gcc
```

```
$ make install
```

2. make xtinstall X-toolkit code

```
$ make xtinstall
```

Install error with X library? Run both commands below.

```
$ sudo apt-get install libx11-dev
```

```
$ sudo apt-get install libxt-dev
```

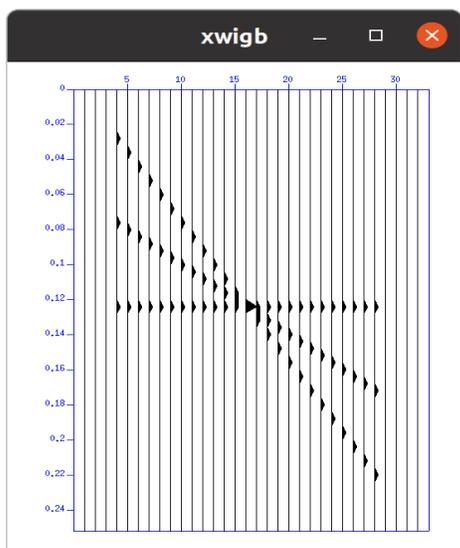
```
$ make xtinstall
```

At this point, with success executing “make install” and “make xinstall,” you can use Seismic Unix. Below, are two commands that yield a wiggle plot (left) and a grey-scale plot (right).

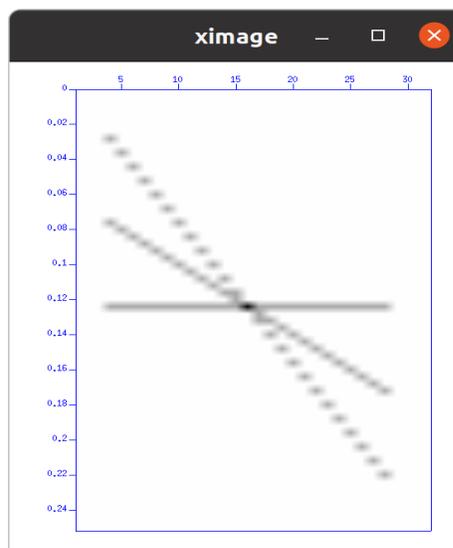
You might develop a preference for one plot type over the other. That’s fine. But, note, when you have a lot of traces to plot, the wiggle plot can be overwhelmed (it will be black). When that happens, try the image plot.

Notice that the two commands below end with “&” (ampersand). The “&” tells the system to run this command as a background process. If you do not “&” at the end of the command, the terminal will be unresponsive – the command is being run in the foreground, it is using the terminal. To stop the process, close the plot window.

```
$ suplane | suxwigb &
```



```
$ suplane | suximage &
```



3. make utils utilities code (optional)

```
$ make utils
```

Error? I do not know! Google for a solution.

4. make finstall FORTRAN code (optional)

```
$ make finstall
```

Install error with FORTRAN?

```
$ sudo apt-get install gfortran
```

```
$ make finstall
```

5. make mginstall Mesa/OpenGL code (optional)

```
$ make mginstall
```

Error?

```
$ sudo apt-get install freeglut3-dev
$ sudo apt-get install libxmu-dev libxmu-headers
$ sudo apt-get install libxi-dev
$ make mglinstall
```

6. make xminstall extra X-Motif code (optional)

```
$ make xminstall
```

Error?

```
$ sudo apt-get update
$ sudo apt-get install libmotif4

$ sudo apt-get install libxm4
$ sudo apt-get install libuil4
$ sudo apt-get install librm4
$ sudo apt-get install libmotif-common

$ sudo apt-get install libxt6
$ sudo apt-get install x11proto-print-dev
$ sudo apt-get install libmotif-dev
$ make xminstall
```

7. make sfinstall SFIO materials and “segdread” (optional)

```
$ make sfinstall
```

Error? I do not know! Google for a solution.

8. autoremove housecleaning (optional)

```
$ sudo apt autoremove
```

Help

Simple program help

To get the simplest program help available, enter the name of the program.

```
$ suwind
```

Program suwind is a windowing program. I have used it to cut off the bottom hundreds of milliseconds of a dataset; for example, to cut off data that are below basement (no good

reflectors). Most of the time, I get all the information I need about a program by this method; that is, by entering the name of a program.

Advanced program help

Another help command with a bit more information is

```
$ sudoc suwind
```

The advantage of this “sudoc” help command is (1) I generally get a bit more information and (2) the end of this screen output usually includes which trace keys (headers) are accessed and modified. When I am developing a complex processing script, this helps me avoid interfering with normal program processing.

Keys (headers) help

There is a very long file that I can access by entering:

```
$ sukeyword -o
```

The command above opens the file and puts me at the beginning of it. I can scroll down the file by entering the space bar (paging) or by pressing the Enter key (line-by-line).

I can also jump directly to a specific key by entering the name of the key; for example,

```
$ sukeyword ns
```

Instead of using this internal documentation, you can open the SU Wiki page “SU Data format”:
https://wiki.seismic-unix.org/sudoc:su_data_format

Bonus discussion: demos directory

Make a backup of the “demos” directories

The **demos** directory has a lot of example scripts to run, to see just how a program runs. But, running a demos script usually change the contents of the directory. The SU instructions tell us to copy a demos directory to another location, then use it there so the original directory is unchanged. I have found that I frequently forget to copy the directory away; therefore, I recommend you make a backup copy of the entire demos subdirectory. I show how to do this below.

```
$ cd /usr/su44r19/src/demos
```

Look around at all the wonderful examples!

Read “README” for an introduction to these directories.

Read “DEMO_STYLE_GUIDE” to understand file suffixes

For safety, make an archive of “demos”

```
$ cd ..                                <-- go up one directory
```

```
$ cp -r demos demos_archive <-- recursive copy
```

Test the recursive copy; that is, test that all subdirectories were copied, not just the top level directory.

```
$ ls demos/Plotting/Psimage
```

```
$ ls demos_archive/Plotting/Psimage
```

The two commands above prove I copied “demos” with its subdirectories because I am able to “ls” directories below “demos.”