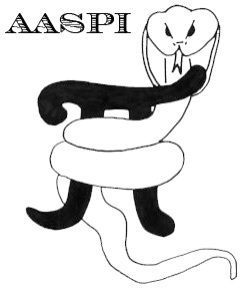


Accessing Linux and Checking your Environmental Variables

Although you may be fortunate enough to have a powerful multi-processor desktop running Linux, most of our sponsors do not. Most of our sponsors will have access to a desktop designed to run seismic interpretation software – either a Linux-based machine perhaps running Seisworks, Voxelgeo, or Geoframe software, or a windows-based PC running Petrel, Geomodeling, Kingdom Suite, or Transform software. In these cases, it may be most efficient (and avoid negatively impacting your local interpretation response time) to run remotely on a Linux cluster or compute server.

Linking to a Linux compute server from a Linux desktop workstation

If you are a first time user of this software and somewhat unfamiliar with Linux, you will want to make sure that your DISPLAY variable is set to run on your local system. Specifically, if you log on to a server called *computer_server.bigoil.com* from your workstation called *local_worstation.bigoil.com* you will want to open a window on your Linux workstation (which may be a window using a Linux emulator such as **PuTTY** and **Xming** from your PC or some other "thin client") and then type:

```
xhost +
```

This command allows anyone to display on your local Linux machine or emulator. Next, secure shell into your compute server by typing:

```
ssh compute_server.bigoil.com
```

Once there, look at the top of your xterm window, which may say something like

```
"your_userid" on local_workstation.bigoil.com.34
```

that indicates window number 34 is opened on your local workstation or emulator. If you are using a Bourne shell, in this window type

```
DISPLAY=local_workstation.bigoil.com:34.0; export DISPLAY
```

Under the bash shell this can be abbreviated as

```
export DISPLAY=local_workstation.bigout.com:34.0
```

In contrast, if you are using a C-shell in your installation, you may need to type

```
setenv DISPLAY local_workstation.bigoil.com:34.0
```

To make sure your DISPLAY is correct, type

```
echo $DISPLAY
```

The 'echo' command should reply with 'local_workstation.bigoil.com:34.0'.

In our environment at OU, we are able to use X-forwarding, which allows the compute-server to display back to the (Linux) area from which you started. If you have this capability, you would simply type

```
ssh –X computer_server.big_oil.com
```

Let's now test your DISPLAY parameters. Type in

```
xterm –bg lightpink &
```

A light pink colored **xterm** should pop up. If you get an error message, it's time to talk to a local IT person for help.

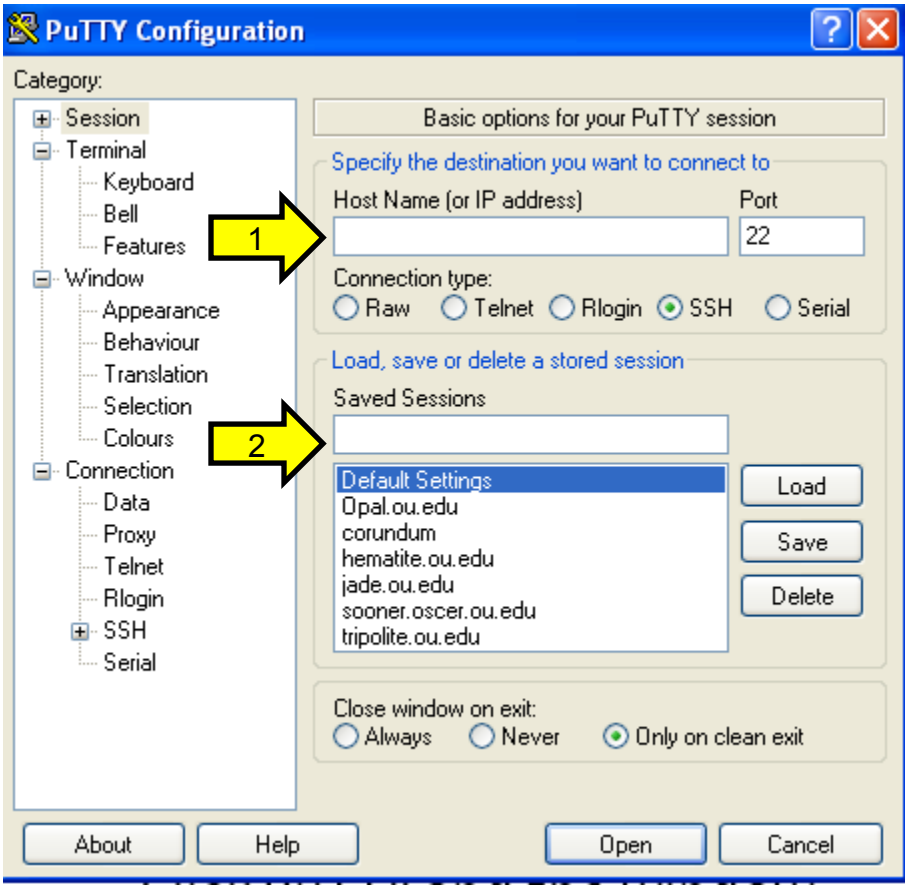
Linking to a Linux compute server from a desktop PC running windows

At OU, all of our compute servers are in a locked, air-conditioned room, with our largest computer server in a different building almost 1 mile away from our desktop PCs. To access a Linux server from your PC, you will want to request your IT folks to download WinSCP and PuTTY or their equivalent. Since they are free, these are kind of software universities use. Don't be worried if your IT folks choose a more expensive communication protocol that better fits into your internal secure network.

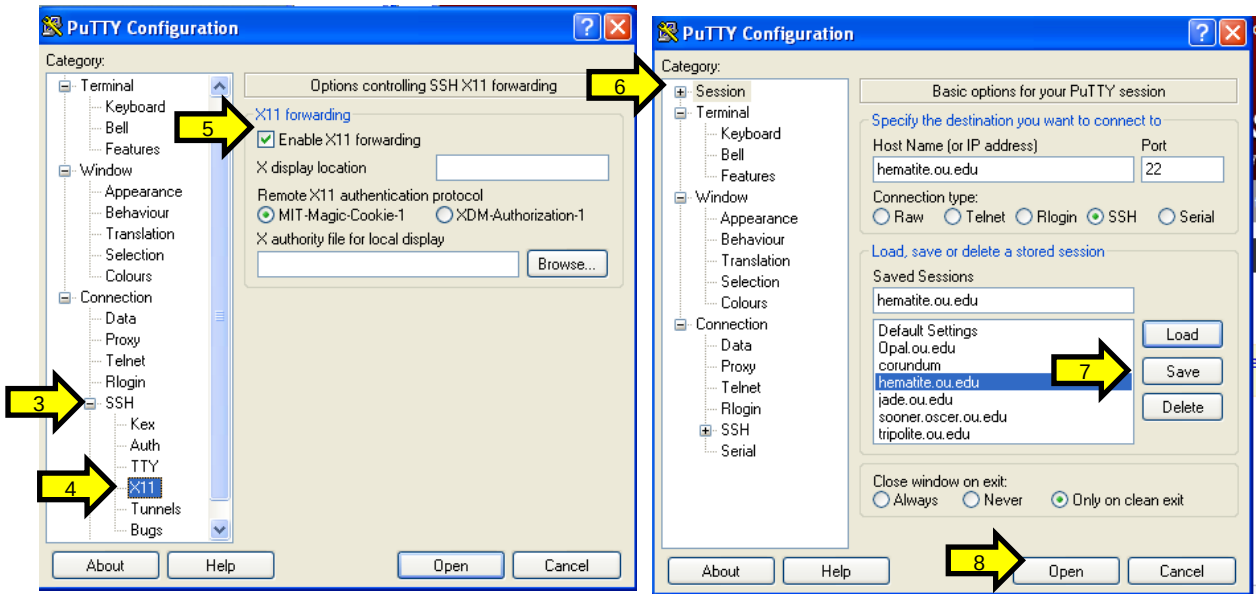
Using PuTTY

After downloading and installing WinSCP and Putty onto my desktop, I see the following two icons:

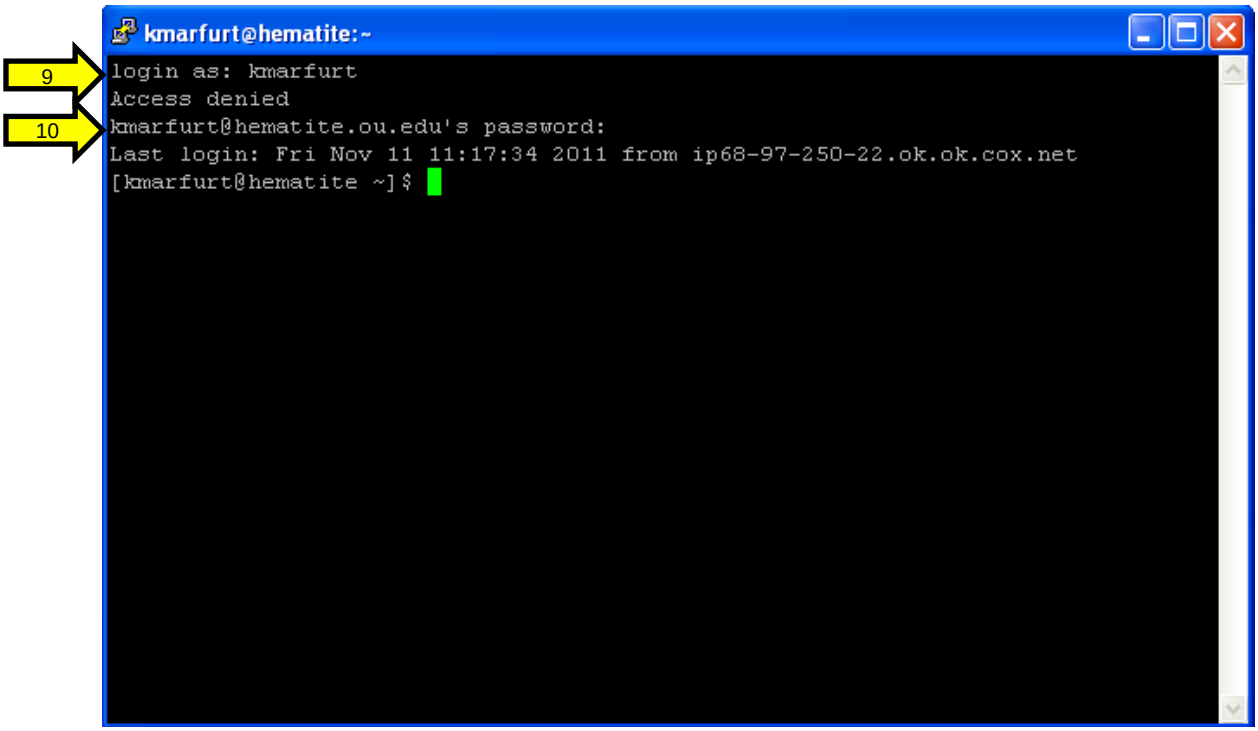
Click **PuTTY** and the window on the right pops up:



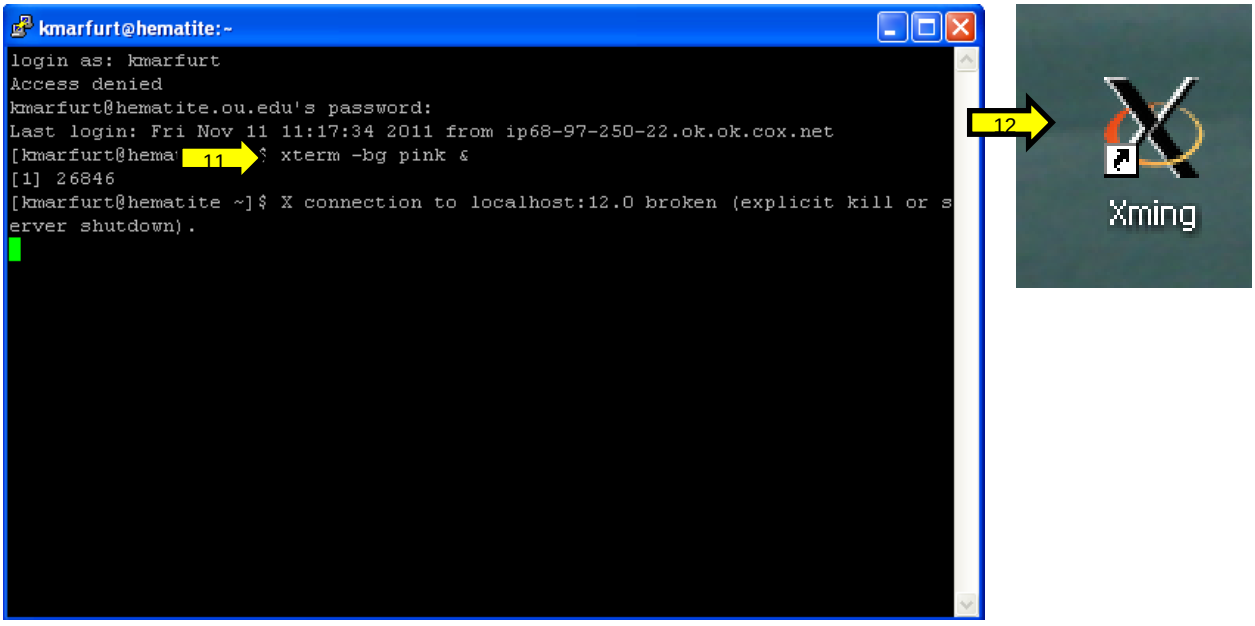
In your first invocation of PuTTY you will have blanks in the Saved Sessions area. At OU, type in 'hematite.ou.edu' under the (1) Host Name and (2) Saved Sessions areas. Don't click Open or Load yet. Obviously, you would type in the name of your local server outside of OU.



Click (3) the SSH box to expand it. Then (4) click **X11**. Then (5) put a check mark in the Enable **X11** forwarding box. This allows any window you open on Linux to be forwarded to the PC from which you invoked **PuTTY**. Next, click (6) Session to return to the original session window. With the name of your saved session (in my case hematite.ou.edu) click (7) Save to save the file name and the **X11** forwarding settings. Finally, click (8) Open to open the **PuTTY** window.

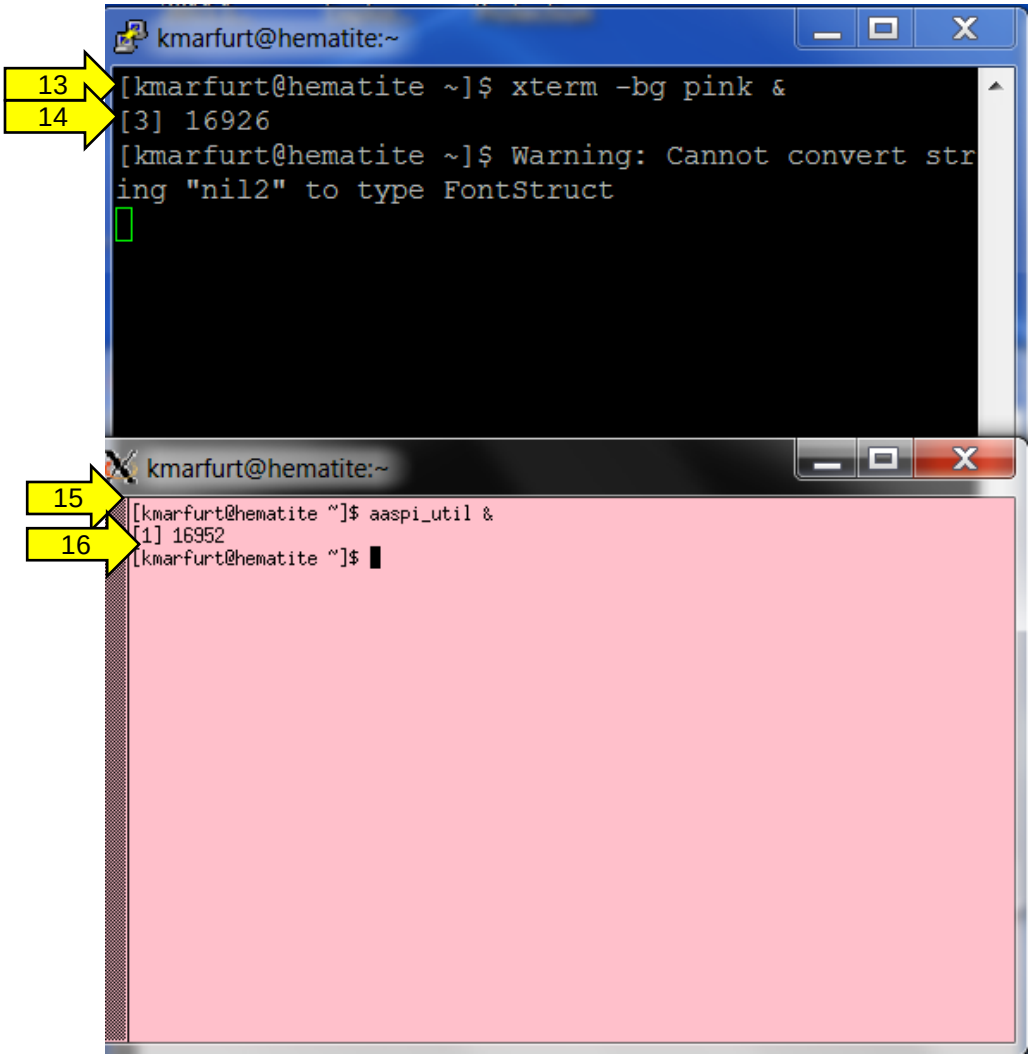


I am (9) prompted for my login user id (in my case it is 'kmarfurt'. Your's will obviously be different. Next (10) I type in my password. I am now in the Linux environment on a machine called hematite.



I type in the Linux command (11) 'xterm –bg pink &' to open up a new X terminal. However, I receive a message that my X connection is broken. This error occurred because I failed to invoke Xming by clicking the Xming icon. Xming runs on the PC.

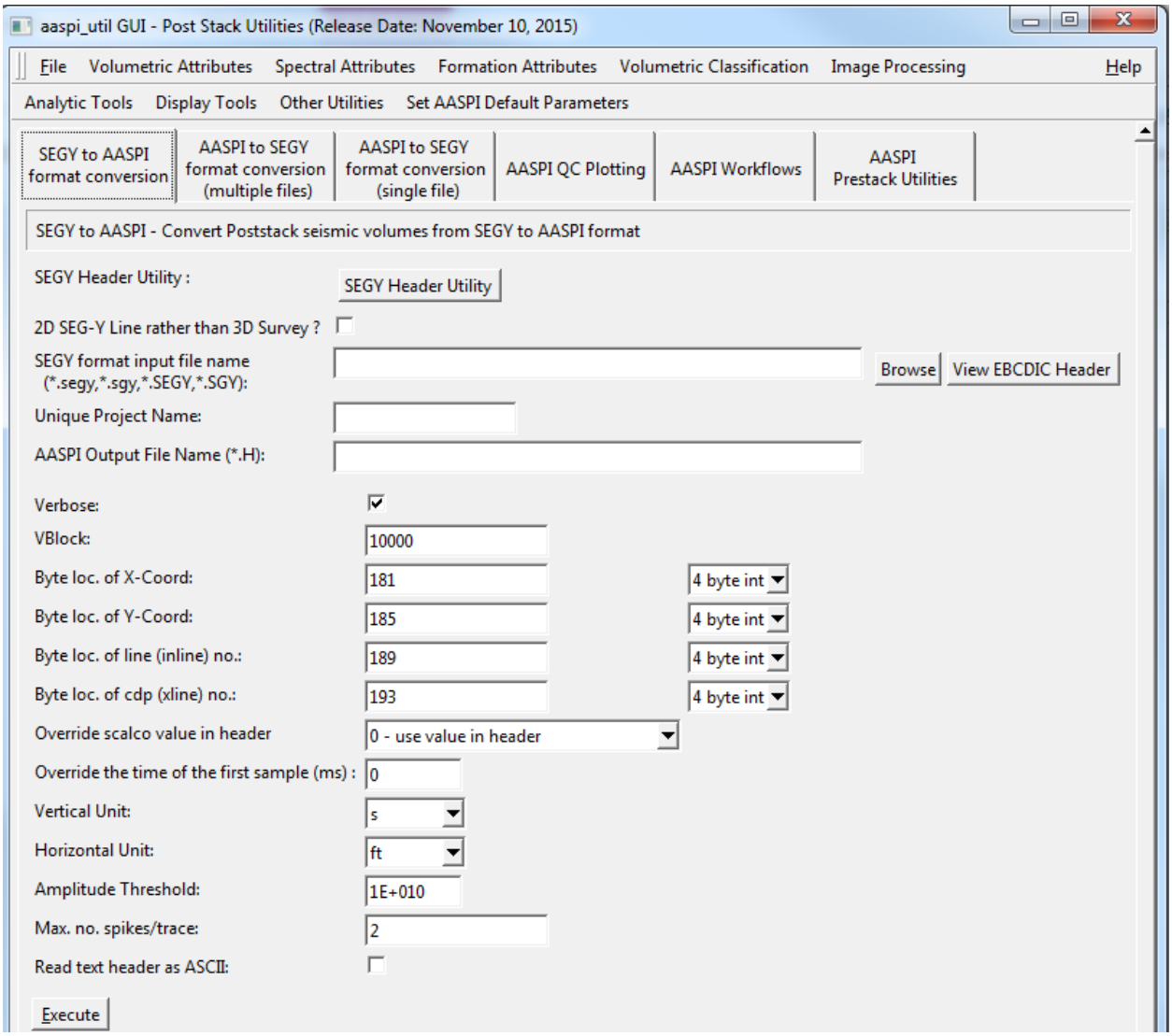
Thus, you will only need to do this once unless you log out of the PC. (12) Click the Xming icon.



I retype in the Linux command (13) 'xterm –bg pink &' . (14) A process id = 16926 appears telling me a new 'job' has been launched. A pink xterm appears. Finally, I invoke the aaspi utility GUI by (15) typing

aaspi_util &

giving rise to a (16) process id = 16952 and the window below:

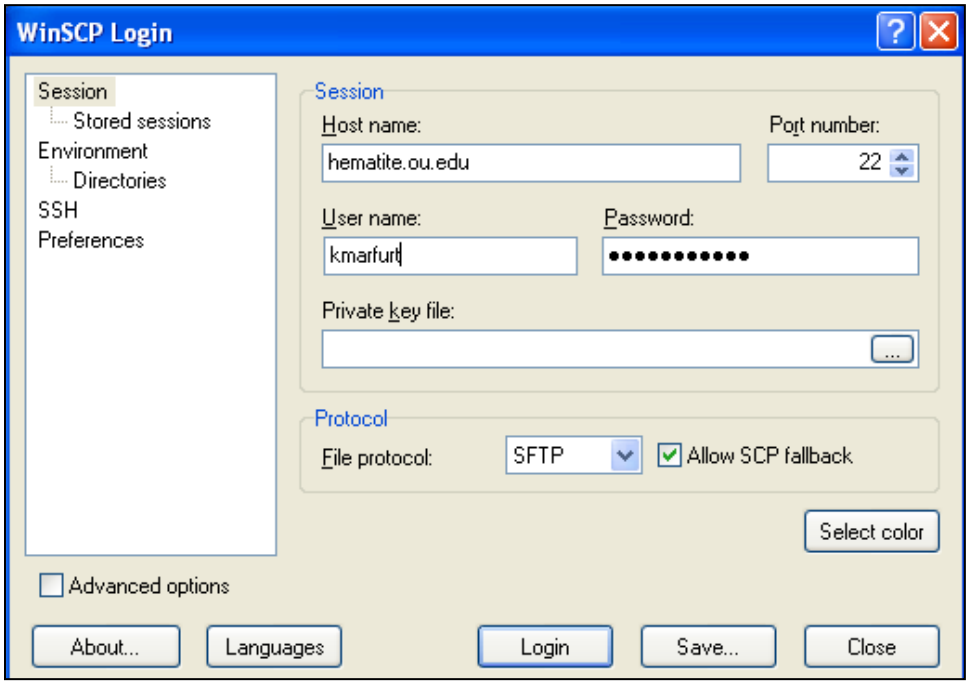


Using the **aaspi_util** GUI will be discussed in different documentations.

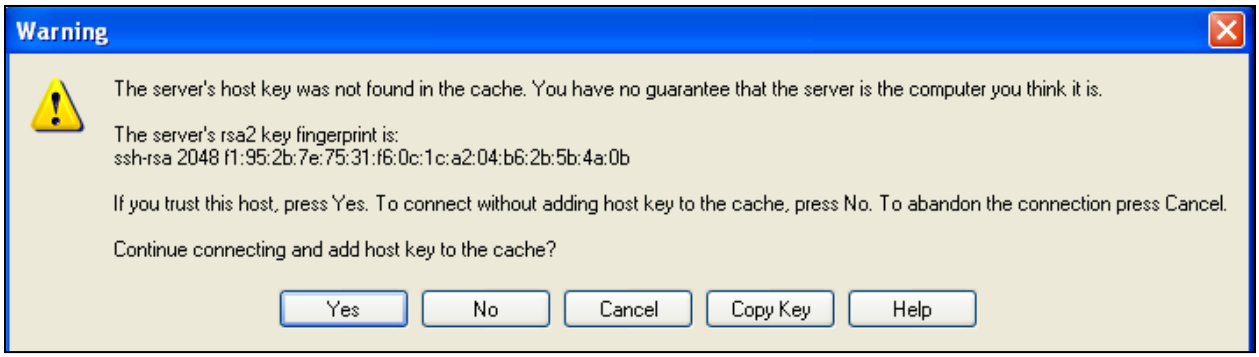
Using WinSCP

To invoke WinSCP, I click the WinSCP icon

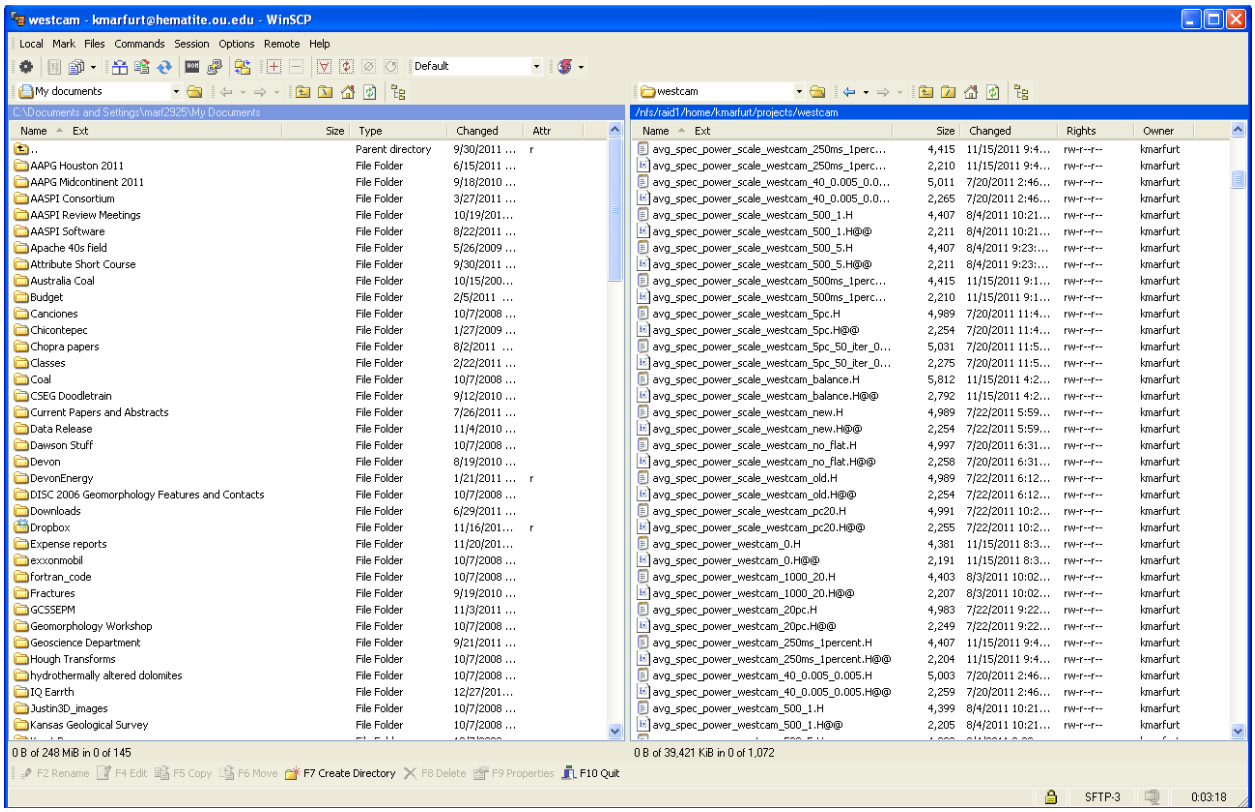
to bring up the pop-up window:



I type in the Host name (hematite.ou.edu) User name (kmarfurt) and Password (xxxxxx), and then click Login. If this is the first time you are connecting to the Linux system from this PC the Warning box below will appear. This prevents malware from connecting you to the North Korean nuclear arsenal.



After connecting the window below appears. The column of files on the left reside in the My documents folder on my laptop. The column of files on the right reside in the westcam directory on the Linux machine. To copy a file from one area to another, simply drag and drop. You can also drag and drop directly to/from a PC folder or file not in the left column, but it may first make a temporary file, slowing the process down somewhat.



Creating a project directory under Linux

Most workers will be assigned a login or data folder in which they can store their work. The first step is therefore to generate a project folder, then one for the Boonsville survey.

Don't forget to create a ".datapath" so that AASPI know where to output file to. If you forget this step, it is highly possible that your home directory (or the /tmp directory) will be flooded and quickly filled by AASPI output files. The best practice for datapath is to output into the current working project directory (i.e. "datapath=./").

If your system reserves a different storage location than your project directory, you can simply move your project directory into the storage location We highly recommend keeping the datapath to be your current working directory (i.e. "./") in order to make Windows and Linux working environment compatible with each other. However, if you wish to keep your project directory location, please read the next section regarding how to define your datapath that is different from your current working directory.

In the Boonsville directory create yet one more directory called ***segy***. Finally copy the seismic segy-format data file into the segy subdirectory. In Marfurt's case, the workflow is simply:

```
cd ~kmarfurt
mkdir projects
cd projects
mkdir boonsville
vim .datapath
(Then type the following:
datapath=./
And save the file then quit)
cd boonsville
mkdir segy
cp /someplace/d_mig_boonsville.segy   segy
ls –ltr segy
```

The final ***ls*** command should echo back the name *d_mig_boonsville.segy* .Once in your project directory, you are now ready to start. In fact, it is the best practice to start aaspi_util from the project directory instead of launching it directly from your home directory, because in that way, all the output files would be in your project directory instead of floating all over your home directory.

Defining a datapath that is different from current working directory – a place in which to store your larger binary seismic data and attribute files

If you wish to store larger binary outputs in a different location than your current working directory, you will need to create a file in your home directory called *.datapath* , where the dot '.' in front of the name indicates that the file will be invisible to Linux program ***ls***. Open this file with your favorite editor (vim, nano, pico, etc.) and type in one line that reads like

```
datapath=/big_disk_drive/my_user_area/AASPI_Data/
```

and save the file. Do not forget the trailing '/' after the path name or you will create files one level up preceded with the letters AASPI_Data. The datapath tells aaspi_io or SEP where to write binary versus ascii files associated with the seismic data. These files can be quite large, such that it may be convenient to put them in a place that gets cleaned up every month or so.

The AASPI software uses aaspi_io-format input, output, display, and command line utilities. The aaspi_io format is a more portable version of the SEP-format developed by members of the Stanford Exploration Project, a geophysical consortium that has been continuously running since the late 1970s. Replacing SEP with aaspi_io utilities enabled us to port the AASPI software to Windows.

I have a file called d_mig_boonsville.H in a project directory. If I edit this file, I note a line that says:

in=/big_disk_drive/my_user_area/AASPI_Data/d_mig_boonsville.H@

that contains the binary seismic samples, and a header format file (hff) descriptor line that reads:

hff= d_mig_boonsville.H@@

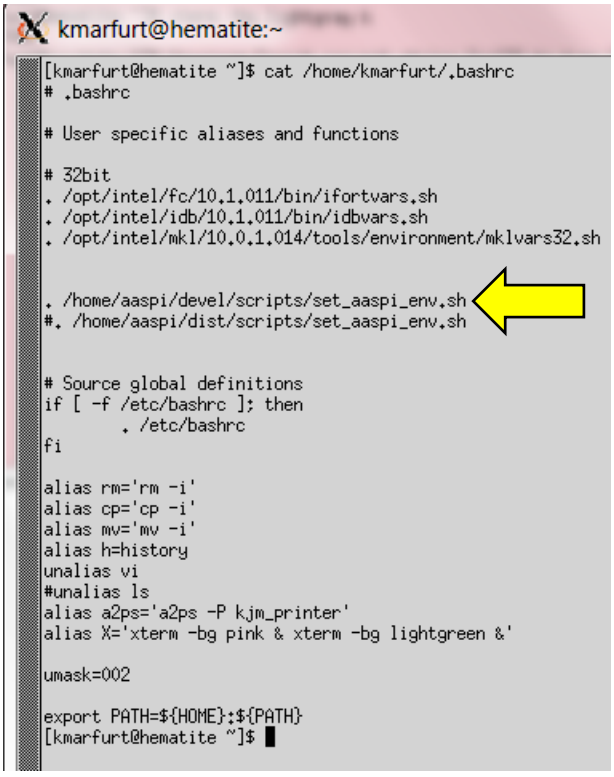
that will be in the local project directory. The *hff* file is also *ascii*. If I edit it, I will see a line that says

in=/big_disk_drive/my_user_area/AASPI_Data/d_mig_boonsville.H@@@

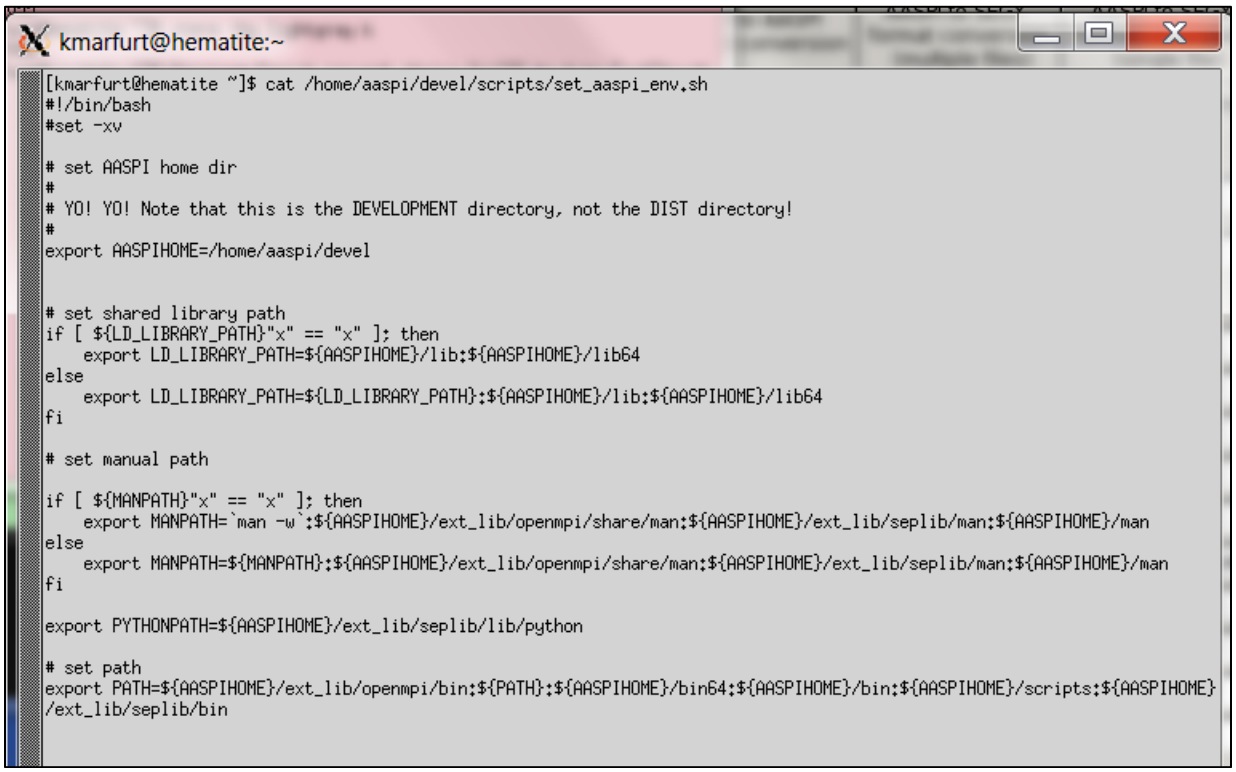
that contains the binary trace headers.

Setting up AASPI Environment Variables

Finally, you need to make sure your PATH and LD_LIBRARY_PATH are set to see the AASPI software, which is achieved by sourcing the shell script *set_aaspi_env.sh*. Ideally, your system admin folks have stuck this into your *.bashrc* file. Editing this script Marfurt sees:



My version of *set_aaspi_env.sh* looks like

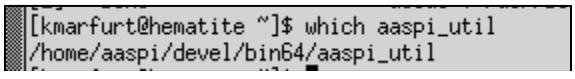


Your local IT person will want to define the variable *AASPIHOME* to the location where he or she has previously loaded the software (in my case it is AASPIHOME=/home/aaspi/devel which is our software development directory. This script should add the AASPI executables, libraries, man pages, and scripts after the current ones set in your Linux environment.

If all is well, you should be able to type:

which aaspi_util

and a long path name pointing to the location of the executable will show up. In the OU environment, it looks like



If your path is incorrect, or if you misspelled the executable name, as shown below, you will get a message like this:

The variables discussed in this section only need to be set once. If you acquire access to a new compute server, or wish to access that compute server from another machine (perhaps your laptop if such access is allowed in your company's environment) you will need to revisit this section again.

Addressing Problems with Pop-up Windows that are too big

Depending on your environment, you may click the 'browse' button on one of the AASPI GUIs and the window that appears is larger than your screen size. It is unclear why this happens, but it is fairly simple to fix. If this occurs, in your home directory, you will find a file called

~/.foxrc/AASPI/GUI

Note the dot "." In front of .foxrc which indicates that it is a "hidden" Linux file, like your .bashrc file.

The contents of my version of ~/.foxrc/AASPI/GUI reads as follows

```
[File Dialog]
height=900
showhidden=0
width=900
style=4194304   .
```

If the width is greater than that of your terminal (mine was width=1300) change it appropriately with your favorite editor to a size smaller than your terminal width (in my case I set width=900).

Next, remove the write permission from this fill so that the fox toolkit does not change it back to width=1300 again (as happened repeatly to me) by typing

chmod -w ~/.foxrc/AASPI/GUI

Once changed, your browser window should fit inside your terminal display.