SAC and AWK Usage and Hints

E.A. Vanacore
April 9, 2007

SAC

Starting SAC: When starting the sac program, first go to the directory that contains the data you wish to analyze then start SAC by typing "sac" in the terminal. You should see a terminal line that looks like SAC>. Note that you must be in sac mode to run sac commands or macros.

Macros: Macros are the same as shells in Unix; they are a series of commands that can be run by typing a single line. This is the same concept as the gmt shells we have used previously in lab. To run a macro in sac, type the command "m *macro_name*".

Some common functions:

r: Read files into sac. The usage for this command is "r *sac_files*"

ppk: This function is used to plot seismograms that have already been read into memory to an X window. The displayed seismograms can been used to examine the data and to record user defined picks. For the purposes of this class, the command line is as follows: "ppk p *#seismograms/plot*"

Some useful ppk window commands.

- x
    - Display the current plot with a new time window (akin to zooming in). The new plot will begin at the current cursor position. Following this marker you must move your cursor to the end of the desired time window and press any key. This will mark the end of the new time window.
- o
    - Display the previous plot window (akin to zooming out). Note that sac only saves up to 5 previous windows.
- b
    - Display the previous plot
- n
    - Display the next plot
- t
    - Define user time Tn in the header (n is 0-9). To mark a time move the cursor to the desire pick, press t, and then press any number 0-9 to save the pick in the header value Tn.

lh: This function lists the header information for sac files already read into memory

wh: This function overwrites the header file information with any current changes made to the header akin to saving an edited text file. This command allows the user to save things such as arrival time picks.

Some sac header definitions
evlo evla evdp: Event longitude, latitude, and depth respectively
stlo stla: Station longitude and latitude respectively
t(0-9): user defined time markers on the seismogram
gcarc: great circle distance between the earthquake and the station (delta) in degrees
delta: sampling rate of the seismogram

AWK

Awk is a programming language used to complete simple data manipulation such as printing reports, finding items, and changing formats. Most programs in this language are one or two lines and can be applied to perform tasks such as creating shells, or creating data tables. In Awk the data is assumed to be stored in columns. In the language individual columns are denoted by a dollar sign followed by a number. For example, if a command contains $4 then the command is calling data from the fourth column of the input file.

Dissecting an AWK Command

The basic awk command has the following format:

Start of commands End of commands
| |
awk '{*commands*}' input_file
| |
Initializes awk awk commands

In the following examples the input file sta.inf is a list of stations and their respective locations (longitude, latitude):

```
AAK.BHZ 74.49440 42.63330
AAM.BHZ 276.343 42.30119
ACCN.BHZ 286.332 43.38430
ACSO.BHZ 277.018 40.23189
ADK.BHZ 183.316 51.88370
AHID.BHZ 248.9 42.76539
AML.BHZ 73.69410 42.13110
```

awk '{print "dumpSHD " $1 " gcarc t8"}' sta.inf > dump-csh

This command prints the following lines into a file called dump-csh. The resultant file will have the following format:

```
dumpSHD AAK.BHZ gcarc t8
dumpSHD AAM.BHZ gcarc t8
dumpSHD ACCN.BHZ gcarc t8
            :
            :
dumpSHD AML.BHZ gcarc t8
```

The command tells the computer to print the first string (dumpSHD) followed by the value of the first column in the file sta.inf and finally print the last string (gcarc t8) for each row in the file sta.inf. Alternatively you can use the command

awk '{print "dumpSHD", $1, "gcarc t8"}' sta.inf > dump-csh

to produce the same output. In this command the comma indicates that there should be white space between each part of the output. Suppose we want to reformat the file sta.inf such that the order of the second and third columns is reversed. To accomplish this we would use the command:

awk '{print $1, $3, $2}' sta.inf > sta_rev.inf

This would produce a file sta_rev.inf that looks like this:

```
AAK.BHZ 42.63330 74.49440
AAM.BHZ 42.30119 276.343
ACCN.BHZ 43.38430 286.332
ACSO.BHZ 40.23189 277.018
ADK.BHZ 51.88370 183.316
AHID.BHZ 42.76539 248.9
AML.BHZ 42.13110 73.69410
```

All of the previous examples require the file sta.inf to create output; we can also pipe in a data result to produce files. For instance if we wish to make the file dump-csh as before without the file sta.inf, then we can pipe the results of an ls command (see Unix_Basics for details) to produce the same output. To do this we use the command:

ls *.BHZ | awk '{print "dumpSHD", $1, "gcarc t8"}' > dump-csh

This command does two things. First it lists the .BHZ files; then it prints the result using the awk command to produce the file dump-csh. The result from the ls command is treated as a single column of data.

We can also pipe the output of an awk command into other commands/scripts. For this lab awk will commonly be piped into gmt commands such as:

awk '{print $2, $3}' sta.inf | psxy –R –Jm –St0.1 –G0/0/255 –K –O >> plot.ps

This command prints the longitude and latitude values for each station into the psxy command that plots the points as blue triangles.

This is just a small sample of what awk is commonly used for. In the scope of lab these basic examples should cover what you need to know. If you are interested in learning more about awk I suggest reading the following book:

The AWK Programming Language by Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger