Basic SeismicHandler Introduction

Sebastian Rost
October 2006

SeismicHandler (SH) is a tool for analyzing digital seismograms. It can be used for the analysis of earthquake records, as well as for examining seismogram sections in refraction seismology. Its strength is the processing and interpretation of seismic array data. Any processing can be applied to multiple traces and the traces can be treated and analyzed as a data ensemble rather than as individual recordings.
SH was developed by Klaus Stammler at the Gräfenberg Array observatory in Germany. It is written in C and can be ported to a variety of computer architectures including Linux PCs.
SH is a command line driven processing tool. There is almost no point and click. This said there is a motif version of SH (called SHM) which includes a lot of pointing and clicking, but since its applications are quite restricted compared to SH, I will not go into this. Command lines can be typed in interactively or may be read from a file (command procedure).

Normally, one will work with two windows; One dialog window for entering command and a second window for graphics output. SH is able to handle up to seven windows but using more than 3 at once is uncommon.

SH contains its own scripting language, which is a mix of C-Shell and C programming language syntax. Using this scripting can help you to speed up your processing and can be used to run batch processing jobs, without (or with) interaction from yourself.

IMPORTANT: SH is not case sensitive and (by default) capitalizes everything you give to it. It also writes out names in capital. Mixtures of numbers and lower case normally do not work! There is a way to use lower case in scripts to use operating system and fortran calls.

Important Files

SH uses special files both in the source directory and in a personal directory under ~/sh/.

Under ~/sh you will find

command/	place to store self written SH scripts. This is the location where SH looks for command verbs. This is likely empty since you did not write any software yet. SH also searches your current directory and the global command directory (see below).
private/	not important
shscratch/	storage space for hardcopies (*.ps) and error messages (*_ERR.STX). Important is SH_LAST.SHC which contains all commands of the last session.

The global directory under $SH_ROOT contains

command/	as above, but a selection of predefined scripts and applications. Browse through, most names make sense and you might get an idea what they do.
doc/	the manual (sh.ps.gz) as gzipped postscript file. Important to read.
filter/	a selection of predefined filters. This and the current directory is the only place SH searches for filters.
globals/	a collections of global variables. No need to touch anything here.
inputs/	collection of station location files and predetermined travel time files

Starting SH

Start-up script called `shn` in `~earsro/bin/`

```
#!/bin/csh
# Call Seismic - Haendlers:
# 04.10.2006 S. Rost Leeds student domain
source /nfs/see-fs-01_u1/earsro/software/sh/setup/shsetup
SH
```

Calling this script sets all the paths and starts up the command line window.

The command line

The command interpreter parses each command line by processing two steps:

1. Split up the command line in words using blanks, semicolons and slashes as terminators. Words are separated either by one or more blanks. Two consecutive semicolons or two semicolons with only blanks in between denote an empty word.
2. Translation of each word.

All words which were separated by a slash ("/") are identified as qualifiers. Qualifiers are treated separately and do not count as parameters. Besides these qualifiers the first word is regarded as the command verb, the others as parameters numbered from 1 to N. N ranges from zero to 15. Empty words result in an empty parameter (or the default value).

Help Pages

There is a limited man-page collection included with SeismicHandler. You can access the list of available help pages with:

```
|sh> help

|sh> help help
command HELP [<cmds>]
============

key: help utility
```

```
Displays help information about seismhandler commands.  It is
possible to get a directory of all commands available (HELP/DIR),
a command menu with short catchwords (HELP/KEY), a list of lines
containing command verb and its parameters (HELP/CALL) or detailed
information about one or more commands specified by <cmds> (without
qualifier).

parameters:

<cmds>  ---  parameter type: string
   Name of command or wild card expression ("*" is wild card for
   arbitrary text).  The HELP command without parameter is equivalent
   to the command "HELP/DIR *", it displays the directory of all
   help items available.

…
```

Not all of the available commands are available, but it helps to read these if you are interested in a specific command.

File Format

SH stores data in two files with the extensions *.QBN and *.QHD. The *.QBN contains the binary data for the individual traces. Since it is binary there are sometimes problems with moving data from big-endian to little-endian machines. The *.QHD file contains the header information for the binary data such as origin time, start time, station name, source location etc. This file is in ASCII.
My file naming convention includes normally the origin time of the earthquake and the array name such as:

09-OCT-2006_01:35_WRA.QBN
09-OCT-2006_01:35_WRA.QHD

Event on October 9, 2006 01:35 recorded at the Warramunga array in Australia. The data you will get is in this format.
For now the specific data format is not important, nor are the possible header variables. For more information check the manual.

Trace Addressing

Many commands require a list of traces as input parameter. Then you will usually specify a number of traces of the current display. The traces are addressed by their position number inside the display window. The positions are counted from the bottom up to the top, starting at 1. By default, the position numbers are displayed on the left side of each trace, where the name of the recording station and the component are given as well.

Important Commands:

`xopen`	:	Open graphics window (try xopen;; to skip the interactive part)
`read name all`	:	Read data in file name (no extension)
`zoom 5 3`	:	Increase display amplitude for trace number 5 by a factor of 3
`markphas`	:	mark arrival time for a set of predetermined phases
`phc`	:	print hard copy – gives PS file – the output filename is being given.
`del all`	:	Erase all traces from screen and memory
`hide 1-3`	:	hides traces one to three from the display. They are kept in memory
`display h:all`	:	gets all traces from memory and displays them in graphics window
`stw`	:	set time window. Cursor driven selection of the time window displayed
`stw 100 200`	:	set time window to 100 to 200 s of data
`dtw`	:	Delete time window
`rd`	:	redraw – Refresh display without doing anything else
`makef`	:	Make filter. Works interactively through a set of options
`fili f <filter>`	:	defines (fourier) filter for filtering
`filter f all`	:	filters data using the defined (fourier) filter.
`write name all`	:	write out data on display into file `name`
`quit y`	:	exit SH

Scripting

Comment lines and comments start with an exclamation mark (“`!`”)

! it is possible to define default values, they start with the word default, then a number and
! then with a default value. What follows after these 3 values is given out as a comment
! when running the code

! default list file with Q-File names
```
default 1 Q_FILES.DAT   q-file list
default 2 1            first file
```

! the cryptic %#1(0) give the value of lines in default value number #1 and defines the
! last value for a while loop in this case
```
default 3 %#1(0)       last file
```

! variable have to be defined using sdef. Up to 21 variables including the default values are
! possible
```
sdef qfile        ! current q-file
```

! this gives the value of the default #2 to the variable filcnt
```
sdef filcnt #2    ! file counter
sdef dec          ! decision
sdef dep          ! depth value
sdef la           ! latitude value
sdef lo           ! longitude value
sdef dist         ! distance vlaue
sdef baz          ! backazimuth value
sdef maxtrc
sdef string
```

! amplitude normalization
```
norm sw
```

! goto jump mark for a if-loop
```
loop_start:
```

! if statement with goto exit condition, “filcnt is the counter
! variables are marked with “name
! integer decision therefore gti
```
   if "filcnt gti #3 goto/forward loop_exit:
```

! gives the value on the filcnt line in the first default file to the variable qfile
! calc s means it is a string variable (i for integer, r for real)

```
   calc s &qfile = %#1("filcnt)
```

! read in the qfile now stored in “qfile
```
   read "qfile all
```

```
! integer calculation $dsptrcs is a built in variable giving the
! maximum numbers on display
calc i &maxtrc = $dsptrcs

! filter definition and filtering
fili f bp_1hz_4hz_6
filter f all

! filtering adds the filtered traces to the display
! so let’s get rid of them

calc s &string = |1|-|"maxtrc|
del "string


! another amplitude normalization
 norm sw

! increasing amplitude on display
zoom all 3

! another if statement with exit mark
! tests if there are any traces on display
 if $dsptrcs eqi 0 goto/forward no_sp_traces:

! this is one of the codes I wrote to mark arrivals on a
! trace – I will provide this since it is a really useful
! tool
mark_taup 1

!this is a way to have an interactive input from a user
! the reply is stored in variable "dec
! “bad event” is printed on the screen
enter &dec bad event?

! string decision therefore eqs
! if good event then something is written out to file
if dec eqs N goto/forward fine_event:
! this set an output channel to the file CHECK.DAT
! then some output and the output channel is closed
echo_ch CHECK.DAT

! header variable are marked with ^
! format descriptors are used like this
calc/fmt=<%6.2@f> r &la = ^lat
echo "qfile "la ^lon ^depth ^distance ^azimuth
echo_ch
fine_event:
```

! this is the goto mark for the no traces on display test
```
no_sp_traces:
```

! cleaning up
! removing all traces from display and memory
```
del all
```

! delete time window
```
Dtw
```

! set the counter up for the next run through
```
calc i &filcnt = "filcnt + 1
```

! go back to the start of the loop
```
goto loop_start:
```

! exit mark for the loop
```
loop_exit:
```

! and this tells SH to return to the command line
```
return
```