BHPIO for Seismic UNIX

BHPIO is a group of SU programs used to read and write SU trace data randomly, using user-specified SU trace headers as keys. When data is written by **BHPWRITE**, up to five trace headers can designated as keys to be used by **BHPREAD** to read the data. The number five is an arbitrary limit, which can easily be changed. In addition to BHPREAD and BHPWRITE, a utility program, **BHPIO**, is used to list summary information about a BHPIO dataset, and to delete BHPIO datasets.

There are currently two distinct versions of BHPIO. One version, the original implementation, is known as the "sequential mode" version, and consists of **BHPWRITESEQ, BHPREADSEQ,** and **BHPIOSEQ**. The newer version, called the "grided mode" version, consists of **BHPWRITECUBE, BHPREADCUBE,** and **BHPIOCUBE**. In addition, there are two scripts, **BHPREAD** and **BHPIO**, which can be used to automatically determine which type of BHPIO dataset the user is requesting, and then run the appropriate version of the BHPREAD or BHPIO.

When using BHPIO to write data, the user must decide which implementation best suits the "type" of data being written, and choose BHPWRITESEQ or BHPWRITECUBE. The main difference between the two is the way in which trace header keys are stored when the data is written. In sequential mode, header keys are saved in an open-ended sequential file for each key. The user does not have to supply any information about the limits of the data being written. In grided mode, header keys are written as an n-dimensional cube, where n is the number of keys the user has specified. Therefore, the maximum number of "bins" or header key slots must be known for each key.

There is advantage and disadvantage for each mode.

Sequential mode advantage: User does not have to know much about the data, which makes it useful in the early stages of processing, when data is being loaded, etc.
Sequential mode disadvantage: Data reading can use significantly more system resources, primarily memory.
Grided mode advantage: Much more efficient than sequential mode.
Grided mode disadvantage: Data limits must be known. However, this is normally the case when dealing with stacked data. Or with pre-stack data which has been through some processing.

Grided mode also offers the ability to hold multiple traces per grid bin, to accommodate "nominal" offsets, etc. In addition, grided mode has some binning rules, which can be used to select output traces.

The self-doc for each BHPIO program follows.

bhpwriteseq < stdin filename=fname [optional parameters]

bhpwriteseq writes a BHPIO "sequential" dataset, which can be randomly accessed by bhpread. The orders in which bhpread can read the dataset are defined by up to 5 trace-header keys which are saved when bhpwriteseq creates the dataset. The sequential term refers to the way in which the trace-header key index is written. The user does not have to have any knowledge about the trace-header values in the data. Contrast with bhpwritecube, which requires the user to define a n-dimensional cube for holding trace header values. bhpwriteseq is useful when the limits of the data being processed are not well defined. However, bhpwritecube is significantly more efficient and uses less system resources than bhpwriteseq.

Required Parameters:

filename=fname	File name

Complete filenames are formed as path/filename_part-num where part-num is a 4-digit partition number

EXAMPLE:

Suppose pathlist= offset-gathers.dat, filename=offset-gathers, and offset-gathers.dat consists of:
/data/D_170_005
/data/D_169_001
/data/D_169_003
Filenames will be /hou/data/D_170_005/offset-gathers_0001,
/hou/data/D_169_001/offset-gathers_0002,
/hou/data/D_169_003/offset-gathers_0003,
/hou/data/D_170_005/offset-gathers_0004,
... until all data are written

Optional Parameters:

pathlist='filename'.dat	ASCII file containing list of paths to use
verbose=0	For debug print
stripe=[yes,no]	Whether to create multiple partitions
size=1999	Size of each file partition(stripe=yes)
key1=fldr	First key
key2=offset	Second key
key3=...	Third key
key4=...	Fourth key
key5=...	Fifth key
endian=native	Omit endian parameter to write native

Specify endian=1 to force BIG_ENDIAN
Specify endian=0 to force LITTLE_ENDIAN

EXAMPLE: Write 3D gathers, saving line, trace, and offset header keys.

```
segyread tape=gathers.sgy |
bhpwriteseq filename=lines3014-3025 key1=ep,R,1 key2=cdp,R,1 \
  key3=offset,I,50
```

Contents of lines3014-3025.dat:

```
 /hou/data/D_169_005/trinidad
 /hou/data/D_170_003/trinidad
```

bhpwritecube < stdin filename=fname [optional parameters]

bhpwritecube writes a BHPIO dataset in "hypercube" format.
The term hypercube refers to the way in which trace-header values
are saved in an n-dimensional cube. The dimensions and size of
the cube are determined by the user's specification of the minimum
value and number of values to save for up to 5 trace headers.
Contrast with bhpwriteseq, which has an "open-ended" method of
defining trace headers to save. bhpwriteseq is useful when the
limits of the data being processed are not well defined. However,
bhpwritecube is significantly more efficient and uses less system
resources than bhpwriteseq.

Required Parameters:

filename=fname	User-supplied filename

Optional Parameters:

init=no	The default action of bhpwrite is to accumulate data and trace index information in existing datasets. This allows you to run multiple jobs that all contributing to the same data volume. Use init=yes to initialize a new dataset.
pathlist='filename'.dat	ASCII file containing list of directories in which to write the dataset. BHPIO datasets are composed of multiple partitions, which can be distributed across multiple UNIX files and filesystems. Names for BHPIO datasets are composed of a path, which comes from the pathlist file, followed by the filename, a partition sequence number, and the extension .su For example, if filename=stack, and stack.dat contains /hou/data/D_170_001 /hou/data/D_170_004 /hou/data/D_170_006 , then complete filenames are /hou/data/D_170_001/stack_0001.su , /hou/data/D_170_004/stack_0002.su , /hou/data/D_170_006/stack_0003.su , /hou/data/D_170_001/stack_0004.su , The size of each file is determined by the size parameter
key1=cdp,1,1,100	
key2=...	
key3=...	
key4=...	

key5=...
Key parameters enable random reading of BHPIO datasets. Up to 5 SU header keys may be specified when a file is created. When you read the file using BHPREAD, it can be read in any order defined by combining the 5 keys which were used to create it. The syntax of the keyn parameter is name,minimum-value,increment,number-of-bins, where name is the SU header name, minimum-value is the minimum header value to write, increment is the increment between successive header values, and number-of-bins is the number of bins to allocate. The total number of traces which can be written for a given key is defined as: ((max-value - min-value + incr)/incr) * trcs-per-bin

bin=1 Maximum traces per bin

action=w Action to take if a key value is less than the specified minimum or greater than the allowable maximum, or if bin overflow occurs. Specify w, i, or a to discard the offending trace and issue a warning, or to discard the offending trace without a warning, or to abort the job, respectively.

rule=keep Binning rule. Choose from:
keep - keep only the first bin=n traces in a bin,
replace - keep only the last trace to occupy a bin.
stack - sum traces into a bin, and update the nhs trace header. The stacked trace is NOT normalized.
min - keep trace with smallest value in binhdr.
max - keep trace with largest value in binhdr.
Rules replace, stack, min and max work only if bin=1

binhdr=offset Binning header, used to decide which trace to keep if rule=min or max

maxens=0 Maximum ensemble size, in traces, in the input data. To switch output partitions whenever the the end of an input ensemble is reached, set maxens to the maximum input ensemble size.
For this option to be effective key1 must correspond to the major sort order of the input data.

prealloc=no Use prealloc=yes to cause output datasets to be filled with zeros before writing any data.

stripe=yes Create multiple partitions of 'size' Megabytes each. Use no to write a single partition. **NOTE**: there is an arbitrary limit of 1 Million traces in a partition.

size=1999 Size of each file partition, in megabytes.

endian=native Omit endian parameter to write native byte-order

Specify endian=1 to force BIG_ENDIAN
Specify endian=0 to force LITTLE_ENDIAN

verbose=0 For debug print

EXAMPLES:

Write 10 lines, each with 101 CDPs of pre-stack MADDOG data:

```
bhpwritecube < line2031-2040.su filename=maddog2031-2040 init=yes \
key1=fldr,2031,1,10 key2=cdp,1500,1,101 key3=offset,181,121,80 bin=2
```

Use bin=2 to keep from losing any offsets.

Same as previous job, but swap partitions on line boundaries.

```
bhpwritecube < line2031-2040.su filename=maddog2031-2040 init=yes \
maxens=80800 key1=fldr,2031,1,10 key2=cdp,1500,1,101 \
key3=offset,181,121,80 bin=2
```

80800 = 10 lines times 101 CDPs times 80 offsets

Initialize dataset with one line of data, then add lines 2-10

```
bhpwritecube < line2031.su filename=maddog2031-2040 init=yes \
key1=fldr,2031,1,10 key2=cdp,1500,1,101 \
key3=offset,181,121,80 bin=2
bhpwritecube < line2032-2036.su filename=maddog2031-2040
bhpwritecube < line2037-2040.su filename=maddog2031-2040
```

Notice that the second 2 jobs specify only the filename.
All other dataset information has already been saved.
The second two jobs can be run serially or concurrently,
but they cannot run until the first job completes.

bhpreadseq filename=fname [optional parameters] > stdout

bhpreadseq reads a BHPIO dataset which has been created by bhpwriteseq. The data can be accessed in order by any combination of the trace headers which were indexed when the dataset was created by bhpwriteseq.

Required Parameters:

filename=fname Base file name; BHPIO will prepend and append file system, etc.

Optional Parameters:

pathlist='filename'.dat
ASCII file containing list of paths

verbose=0 For debug print

request=data data to get data, or summary to get the following information about a data request: min,max,mean,rms values, samples-per-trace,number of traces, sample units(0=time,1=feet,2=meters), sampling interval, start time

keys=fldr,offset Up to 5 header keys

keylist=p1,p2:s1-s2 List, ranges to read for each key, separated by colon; Valid syntax includes:
k1,k2,... List of specific keys
k1-k2 Range
k1-k2[k3] Range, with increment
k1^k2 Vector, project line from k1 to k2, and read traces nearest the line
* Read all

If keylist is specified, it must contain an entry for each specified key

nearest=0,0 Comma-separated flags, one flag for for each key; 1=take nearest key if value in keylist not found; 0=take only values which match keylist

If nearest is specified, it must contain an entry for each specified key

combine=1 Number of primary key ensembles to combine for supergathers

binsize=k1,k2 Binsize for each specified key

timeslice=st,end,inc Timeslice start time, end time, incr

output=$TMPDIR/$USER/timeslice
Directory to write timeslices

display=yes Display time slices; no=don't display

interpolate=bilinear Spatial interpolation option
Requires primary and secondary keys
Interpolates trace values
Default is bilinear interpolation
Use interpolate=nearest to take nearest
trace; this requires the data
to be in a requlat 3D grid, in which
the primary, secondary key values
are the nodes of the grid

endian=my_endian Omit endian parameter to write native
Specify endian=1 to force BIG_ENDIAN
Specify endian=0 to force LITTLE_ENDIAN

EXAMPLE: Display a gather from previously created gathers dataset

```
bhpreadseq filename=lines3014-3025 keys=ep,cdp keylist=3020:1400 | suximage
```

bhpreadcube - Read bhpio dataset

bhpread filename= [optional parameters]

Required Parameters:

filename=	File name which was used by BHPWRITE to create the dataset.

Optional Parameters:

pathlist= 'filename'.dat	ASCII file containing the same list of paths which were used with BHPWRITE to create the dataset
keys=fldr,offset	Up to 5 header keys which were used to create the dataset.
keylist=p1,p2:s1-s2	List of trace header values to read for each key. Use a colon to delimit the list for each key. Valid syntax includes: k1,k2,... List of specific keys k1-k2 Range k1-k2[k3] Range, with increment * Read all

BHPREAD passes traces in the order specified by keylist.
If keylist is specified, it must contain an entry for each
key in the keys parameter. If keys are not specified, BHPREAD passes
traces in the order they were written by BHPWRITE.

rule=near	Trace selection rule: near, match, all, or stack rule=near - select trace nearest requested one rule=match - select a trace only if it matches the request rule=all - select all traces in the bin rule=stack - stack all traces in the bin Stack is normalized by the fold count of each sample, and header nhs has number of summed traces.
maxtraces=0	Maximum number of traces to read. Zero means read all requested data. If maxtraces is specified, and the specified keylist results in more than maxtraces, output will be truncated.
request=data	Default action is to pass traces out. request=summary will pass information about the requested data instead. For request=summary, the following information will be written to stdout: min, max, mean, and rms values of the

	requested data, samples-per-trace, number of traces, sample units(0=time,2=depth) time/depth interval, start and end time/depth
endian=my_endian	Omit endian parameter to write native format. Specify endian=1 to force BIG_ENDIAN Specify endian=0 to force LITTLE_ENDIAN
foldfile=	To produce an ASCII fold count file. No traces will be written out. An ASCII file containing primary-key,secondary-key,fold-count will be written to foldfile
verbose=0	For debug print, use verbose=1

EXAMPLES:

Read 1 line from the 10-line maddog prestack volume:

```
bhpread filename=maddog2031-2040 keys=fldr keylist=2035 rule=all \
> 2035.su
```

NOTE: rule=all is used to get all the offsets.

Read a crossline:

```
bhpread filename=maddog2031-2040 keys=cdp keylist=1550 rule=all
> 1550.su
```

Create near-offset gathers:

```
bhpread filename=maddog2031-2040 keys=fldr,cdp,offset \
keylist=*:*:1-750 > near.su
```

The calling sequence for both versions of BHPIO is the same.

BHPIO is a utility program used to display a summary of a BHPIO dataset, and optionally, to delete the dataset.

bhpio filename= [optional parameters]

Required Parameters:

filename=	File name which was used to create the dataset

Optional Parameters:

pathlist=	'filename'.dat Pathlist file which was used to create the dataset
verbose=0	Use verbose=1 for debug print
delete=no	yes/no to delete an existing file