Chandra X-ray Center

URL: http://cxc.harvard.edu/sherpa3.4/bugs/sl_sherpa_eval.html
Last modified: 21 September 2006

S−lang−related Bugs: sherpa_eval

Bugs

1. ***sherpa_eval("lplot <item does not exist>")* causes a segmentation fault. (Mac OS X)**

   e.g. "lplot resid" when no source model is defined. The crash occurs only when a script is evaluated by slsh, and it occurs when the script is finishing, after completing all the commands.

   There are other commands that cause this problem; the following is not guaranteed to be a complete list.

   - create_model()

     The script

     ```
     import("sherpa");
     () = create_model( "const2d", "bgnd" );
     message( "Finished." );
     ```
     will − when run by slsh on OS X − Seg fault after printing the message "Finished.".

     Workaround: use `sherpa_eval()` instead, so in this example it would be

     ```
     import("sherpa");
     () = sherpa_eval( "const2d[bgnd]" );
     message( "Finished." );
     ```
     The bug does not appear to be related to the actual model being created.
   - get_par("modelname")

     `get_par()` will cause this crash on exit if it is called with the name of the model. It does not crash when called with either no arguments or with the name of a parameter: e.g.

     ```
     import("sherpa");
     () = sherpa_eval( "polynom1d[pl]" );
     variable pars = get_par( "pl" );
     message( "Finished." );
     ```
     crashes after printing out "Finished." The ***workaround*** to either call `get_par()` and loop through the returned parameters looking for those whose name matches the model you want, or find out all the parameters for the model of interest and then run `get_par()` on each parameter name. For instance the function

     ```
     define get_model_pars(mname) {

       variable pars = Assoc_Type [Struct_Type];
     ```

```
  foreach ( get_par() ) {
    variable par = ();

    % skip those whose model does not match mname
    variable names = strtok( par.name, "." );
    if ( 0 != strcmp( mname, names[0] ) )
      continue;

    % could use names[1] instead of par.name to just use the
    % name of the parameter (i.e. no model name)
    %
    pars[par.name] = par;
  }
    return pars;
}
```

will return an associative array whose keys are the parameter names of the given model. So, using the example above

```
  variable p = get_model_pars("pl");
```

would return an associative array with keys of "`pl.c0`", "`pl.c1`", etc.

2. ***`xspec abundan`* always returns a status of 0.**

   Executing `xspec abundan` by means of `sherpa_eval` returns 0 even when given an incorrect argument. For example, the following commands work (i.e. return 0 on success and −1 on failure):

```
sherpa> sherpa_eval( "xspec abundan angr" )
    Abundances set to Anders & Grevesse
0
sherpa> sherpa_eval( "xspec abundan" )
−1
```

   However the following do not recognize a failure:

```
sherpa> sherpa_eval( "xspec abundan file foo" )
Failed to open requested file with abundances
0
sherpa> sherpa_eval( "xspec abundan X" )
 'X' does not match.
  Choose from the following ABUND options (currently 'file'):
 angr feld aneb grsa wilm lodd file
0
```

3. ***`sherpa_eval("data ...")`* causes a segmentation fault.**
   (Mac OS X)

URL: http://cxc.harvard.edu/sherpa3.4/bugs/sl_sherpa_eval.html
Last modified: 21 September 2006