

LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

| |
|---|
| Document Type LIGO-T01xxxx-00 - E Nov. 1997 |
| e2ecli Command Line Interface |
| Hiro Yamamoto |

Distribution of this draft:

xyz

This is an internal working note
of the LIGO Project..

California Institute of Technology
LIGO Project - MS 51-33
Pasadena CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project - MS 20B-145
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

WWW: <http://www.ligo.caltech.edu/>

LIGO DRAFT

1 ABSTRACT

This document describes the cli, Command Line Interface, package used in the End to End model simulation package. This package replaces the interaction with the user using the standard input and output stream “cin” and “cout”. This makes it easy to use the program in various ways. E.g., the user can record all key strokes in a file and play them back later, or the user can see an explanation about the prompt.

2 KEYWORDS

End to End model, Adlib, cli, auto recording, playback, help

3 OVERVIEW

In this section, the overview of this e2ekli package is explained for the typical three cases, (1) getting one value, (2) getting multiple related values and (3) getting selection of names in a given list. Also covered in this section at the end is how to record the key strokes automatically and playback them later, and about the comments in the recorded file.

3.1. getting a value - getval

Usually, a program gets input from the user in the following way using the standard output and input stream:

```
cout << "Enter the start frequency >> " << endl;
cin >> start_freq;
```

and the user who runs the program sees the following prompts, and types the value and hit return:

```
Enter the start frequency >>
```

This program code will be replaced by the following line:

```
start_freq = e2ekli_getDbl(      "Enter the start frequency", // message
                                "Simulation starts from this frequency", // help
                                1.0, // default value
                                1.0e-5, 1.0e5 ); // range
```

The user sees the following line:

```
E2E> Enter a double (def=1,[1e-05:100000]) >>
```

The first argument of `e2ekli_getDbl` is the prompt, and the second argument is the help message which is displayed when the user types “?” as follow.

```
E2E> Enter a double (def=1,[1e-05:100000]) >> ?
E2E> Simulation starts from this frequency
```

The third argument is the default value, which is shown in the prompt message as “def=1”. If the default value is provided, the user can type carriage return to accept the value. The fourth and fifth arguments are the lower and upper limit of the acceptable value, which is shown as “[1e-5,100000]” in the prompt. If the user enter a value of out this range, the following message is printed and the user is prompted for another value.

```
E2E> Enter a double (def=1,[1e-05:100000]) >> 0
%ERR>
```

```
%ERR> value (0) is out of limits [1e-05:100000]. Please enter a value in limits.
%ERR>
E2E> Enter a double (def=1,[1e-05:100000]) >>
```

The default, lower limit and upper limit are optional, and if the help message is a zero length string, “no help available” is printed when “?” is entered.

There are variants of `e2ecli_getDbl`. `e2ecli_getInt` is to get an integer value, `e2ecli_getStr` to get a string and `e2ecli_getBool` to get a boolean value. To answer the question of `e2ecli_getBool`, one can use “1”, “true”, “yes” for a positive answer, and “0”, “false”, “no” for a negative answer.

3.2. getting multiple related values - modval

There are cases when a program needs to get related values, e.g., various parameters defining the shape of a detector. Usually, a user is presented one prompt after another. By using `modval` routines of `e2ecli`, the user is prompted with related parameters all at once.

The code examples to ask the shape of the a detector is given below:

```
// modval object which handles double variables
e2ecli_dbl_modval mv;
// add 4 double variables to modval
mv.addItem( "r_min", "Inner radius of the detector in units of spot size", 0, 0 );
mv.addItem( "r_max", "Outer radius of the detector in units of spot size", 5, 0 );
mv.addItem( "phi_begin", "Start of the azimuthal angler in degree", 0, -180, 360 );
mv.addItem( "phi_end", "End of the azimuthal angler in degree", 360, 0, 360);
// interact with the user
mv.handler("Define the detector shape" );
// copy the values to local variables
r_min = mv.getVal( 0 );
r_max = mv.getVal( 1 );
phi_begin = mv.getVal( 2 );
phi_end = mv.getVal( 3 );
```

The arguments of `addItem` are the same as those of `e2ecli_getDbl`. `handler` member function takes care of the interaction with the user. After the user finish the data entry, the values can be retrieved by `getVal(index)`, where `index` is the order the item is `addItem`'ed.

The user sees the following prompts when `handler` is called:

```
E2E> Define the detector shape
E2E>
E2E>   r_min      [0:INF]      = 0
E2E>   r_max      [0:INF]      = 5
E2E>   phi_begin  [-180:360]   = 0
E2E>   phi_end    [0:360]      = 360
E2E>
E2E> "name = val", "?" for help or OK >>
```

The right hand side of each “=” is the current value of the parameter on the left hand side, and the values in “[]” shows the range of the parameter, and one can see the help message by typing “?”.

```

E2E> "name = val", "?" for help or OK >> ?
E2E>
E2E>   r_min      : Inner radius of the detector in units of spot size
E2E>   r_max      : Outer radius of the detector in units of spot size
E2E>   phi_begin  : Start of the azimuthal angler in degree
E2E>   phi_end    : End of the azimuthal angler in degree
E2E>

```

The user can change the values of each parameter by typing “parameter name = val”:

```

E2E> "name = val", "?" for help or OK >> phi_b=45, phi_e=135
E2E>
E2E> Define the detector shape
E2E>
E2E>   r_min      [0:INF]      = 0
E2E>   r_max      [0:INF]      = 5
E2E>   phi_begin  [-180:360]   = 45
E2E>   phi_end    [0:INF]      = 135

```

There are a few points to note. The string specifying a parameter does not need to be the full string, but it only needs to be long enough to identify the parameter uniquely. E.g., in the above example, in order to give a new value to `phi_begin`, one can type “`phi_b=45`”, because `phi_b` is long enough to distinguish `phi_begin` from other names. Second point is that one can place several commands in one reply using “,” as a separator.

When the appropriate values are assigned for all parameters, the user has to type “ok” to accept the data entry to proceed forward.

```

E2E> "name = val", "?" for help or OK >> ok

```

3.3. getting selection of names

Third case is to ask the user to make a selection of items from a given list of names. Examples are (1) choose the shaker point of swept sine measurement from a given list of input sources and (2) output port names to be analyzed from a given list of output port names.

The following is a source code asking the user to select output ports to be analyzed.

```

// inquire object
e2ekli_inquire inq;
// add name to the object
inq.addItem("InDemod", "Inphase demodulated error signal", true );
inq.addItem("QuDemod", "Quadphase demodulated error signal", true );
inq.addItem("Power", "Power in the arm cavity", false );
// interaction with the user
inq.handler( "Select output ports to be analyzed", 1, 2 );
// get the index list of selected items
vector< size_t > selected;
inq.get_selection( &selected );

```

LIGO-DRAFT

The arguments of the `addItem` function are the name of the parameter, the help text and a boolean value if the parameter is to be selected to not by default. `handler` takes care of the interaction with the user. The second and third arguments are minimum and maximum number of selections required. In this example, there should be one or two selections. `get_selection` returns a vector of integer which keeps the indexes of the selected items.

With the example source, the user will see the following prompt:

```
E2E> Select output ports to be analyzed
E2E>
E2E>      *InDemod      *QuDemod      Power
E2E>
E2E> Select 1 to 2 items
E2E> Type "name" to toggle on/off  or OK >>
```

The item selected is marked by “*” preceding the name. If the user types just a name in the list without on or off, the selection status toggles, i.e., a selected item is deselected and an unselected item is selected. The user can type “parameter name = true” or “parameter name = false” to define the new status irrelevant of the current status.

```
E2E> Type "name" to toggle on/off  or OK >> In=false, Power=true
E2E>
E2E> Select output ports to be analyzed
E2E>
E2E>      InDemod      *QuDemod      *Power
E2E>
E2E> Select 1 to 2 items
E2E> Type "name" to toggle on/off  or OK >>
```

The help works in the same way as others, i.e., if the user types “?”, the help texts are printed.

```
E2E> Type "name" to toggle on/off  or OK >> ?
E2E>
E2E>      InDemod : Inphase demodulated error signal
E2E>      QuDemod : Quadphase demodulated error signal
E2E>      Power   : Power in the arm cavity
E2E> Type "+"("-") to turn all on (off)
E2E> or type "name" = on or off to force to be on or off.
E2E>
E2E> Select 1 to 2 items
E2E> Type "name" to toggle on/off  or OK >>
```

When appropriate number of items are selected, one can type “ok” to proceed to next.

```
E2E> Type "name" to toggle on/off  or OK >> ok
```

If you do not specify the number of selections by the second and third argument of the handler function, it is equivalent to using a range from 0 to the number of items. If you specify one value for the number, i.e., if you call

```
inq.handler( "Select output ports to be analyzed", 1 );
```

Then when the user types a valid name, the program gains the control back. This is useful when you ask the user to choose an action from a list of choices, like “what do you want to work on, SUS/SEI, IOO, COC, LSC or ASC ?” When according to the answer, you present parameters appropriate for the choice.

3.4. key entry recording and playback - @(fname, @) and @fname

At any prompt by the e2ecli interface routine, one can type

```
@(filename
```

to start recording all the key strokes in the file “filename”. Either at the end of the program or when the user types

```
@)
```

the recording ends. One can play back this recorded key stroke by typing

```
@filename
```

E.g., if the user types in the following way, a file named dbl.cmd is created.

```
E2E> Enter a double (def=1,[1e-05:100000]) >> @(dbl.cmd
E2E> Enter a double (def=1,[1e-05:100000]) >> 123
E2E> Enter an integer (def=10,[0:100]) >> @)
```

The content of the file is as follows:

```
% Enter a double (def=1,[1e-05:100000]) >>
123
% Enter an integer (def=10,[0:100]) >>
```

When this file is used to playback the key entry later, the “%” denotes a beginning of a comment, and the rest of the line is neglected. So this file effectively contains only one line, “123”.

When the program generates many prompts and confirmations in the interactive session, it may be clumsy to see them all replayed when this reply feature is used. In order to suppress those interactions, insert

```
%PROMPTOFF
```

Then all outputs from e2ecli are suppressed until the line which contains

```
%PROMPTON
```

4 USING E2ECLI

This chapter explains the user experience in programs which use the e2ecli package.

4.1. basics

```
E2E> Enter an integer (def=10,[0:100]) >>
```

This is a prompt users see when an integer is to be

```
E2E> Enter an integer (def=10,[0:100]) >> ?
```

```
E2E> Help message about this int
```

```
E2E> Enter an integer (def=10,[0:100]) >> -1
```

```
%ERR>
```

```
%ERR> value (-1) is out of limits [0:100]. Please enter a value in
limits.
```

```
%ERR>
```

```
E2E> Enter an integer (def=10,[0:100]) >>
```

5 REFERENCE

xxx

6 SUMMARY

xxx

APPENDIX 1 APPENDIX TITLE

xxx

APPENDIX 2 ANOTHER APPENDIX

xxx

LIGO-DRAFT